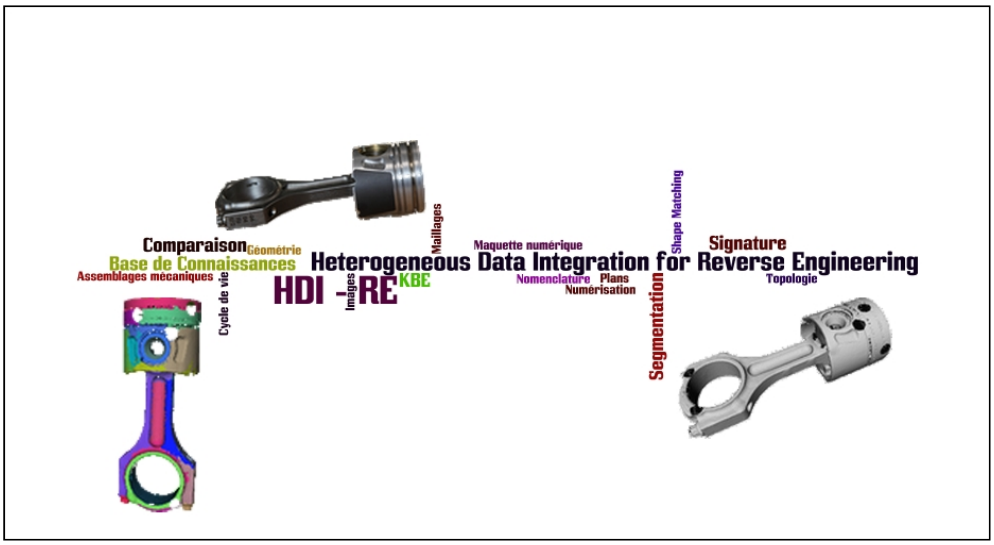
Une méthodologie de Reverse Engineering à partir de données hétérogènes pour les pièces et assemblages mécaniques

Marina Bruneau

Par **Marina BRUNEAU**

Une méthodologie de Reverse Engineering à partir de données hétérogènes pour les pièces et assemblages mécaniques

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 22 mars 2016
Spécialité : Mécanique Avancée

D2267

Une méthodologie de Reverse Engineering à partir de données hétérogènes pour les pièces et assemblages mécaniques

Marina BRUNEAU

Thèse soutenue le 22 mars 2016 devant le jury composé de :

Président :

Jean-Philippe PERNOT
Professeur des universités
Arts et Métiers ParisTech campus d'Aix-en-Provence

Rapporteurs :

Christian MASCLE
Professeur titulaire Dpt de génie mécanique
Polytechnique Montréal (Canada)

Claire LARTIGUE
Professeur des universités
IUT de Cachan

Examinateurs :

Matthieu BRICOGNE
Maître de Conférences
Univ. de Technologie de Compiègne

Gilles GESQUIERE
Professeur des universités
Université Lumière Lyon 2

Membres invités :

Harvey ROWSON
Responsable Projets
DeltaCAD, Lacroix Saint-Ouen

Sándor VAJNA
Professeur des universités, Dr.-Ing., Docteur h. c.
Université Otto-von-Guericke (Allemagne)

Directeurs de Thèse :

Alexandre DURUPT
Maître de conférences
Univ. de Technologie de Compiègne

Lionel ROUCOULES
Professeur des universités
Arts et Métiers ParisTech campus d'Aix-en-Pce

Université de Technologie de Compiègne

Laboratoire Roberval UMR CNRS 7337

22 Mars 2016

Résumé

Cette thèse traite d'une méthodologie de Reverse Engineering (RE) d'assemblages mécaniques à partir de données hétérogènes dans un contexte routinier. Cette activité consiste à partir d'un produit ou d'un assemblage, à récupérer la donnée numérique en partant de la donnée physique dans le but de reconstruire sa maquette numérique. Plusieurs techniques de numérisation peuvent être employées et permettent de générer des données de différents types (ex: nuage de points, photographies). Ces dernières sont utilisées comme données d'entrée à notre processus de RE et peuvent aussi être associées à des données liées au produit, existantes au préalable, telles que des mises en plan ou encore une version antérieure de la maquette numérique du produit. Le traitement de l'ensemble de ces données, dites "hétérogènes", requiert une solution qui soit capable de gérer d'une part, l'hétérogénéité des données et des informations qu'elles contiennent et d'autre part, l'incomplétude de certaines données qui est liée au bruit ou à la technologie utilisée pour numériser l'assemblage (ex : scanner ou photographie). Enfin la pertinence des informations extraites lors de la phase de traitement doit permettre, dans certains cas, de générer des modèles CAO paramétrés, propres à l'activité de RE de l'entreprise ou du domaine d'application.

L'état de l'art sur la reconnaissance de formes dans des données hétérogènes ainsi que sur la gestion de connaissances dans le cadre d'activités routinières, propose des approches qui traitent soit d'un seul type de données, soit du RE de pièce unique ou soit elles ne permettent pas d'obtenir un modèle CAO qui soit exploitable (paramétrage géométrique des entités) pour une activité de RE.

Cette thèse propose une méthodologie nommée **H**eterogeneous **D**ata **I**ntegration for **R**everse **E**ngineering (HDI-RE) et qui se décompose en trois étapes : la segmentation, la signature et la comparaison avec une base de connaissances. Le but de cette méthode est d'automatiser le processus de RE et notamment en ce qui concerne les étapes de reconnaissance de composants dans les données d'entrée et d'aide à la reconstruction de modèles CAO (paramétrés ou non) en récupérant des informations géométriques et topologiques dans les données d'entrée. Pour cela, ces dernières sont segmentées afin d'en extraire des informations qui sont ensuite formalisées sous la forme de signatures. Les signatures générées sont ensuite comparées à une base de connaissances comportant un ensemble de signatures de différents types et appartenant à des produits ou objets déjà connus. Le calcul des similarités issu de la comparaison permet d'identifier les composants présents dans les données en entrée.

L'apport scientifique de ces travaux repose principalement sur l'utilisation de signatures qui, en fonction du souhait de l'utilisateur, permettent de reconstruire une maquette numérique en sortie du processus de RE avec trois niveaux d'information : un niveau global, un niveau géométrique et topologique ou un niveau fonctionnel. Par rapport à chaque niveau et du type de données traité, un mécanisme de signature dédié est proposé.

à mon père

Remerciements

Je voudrais tout d'abord remercier Madame Claire Lartigue et Monsieur Christian Mascle d'avoir accepté de rapporter ces travaux. Je tiens également à remercier mes deux directeurs de thèse, Alexandre et Lionel, pour ces trois années de thèse au cours desquelles ils m'ont transmis leurs connaissances et ont su me guider tout au long de ces recherches. Je les remercie pour leur patience, leur écoute et leurs conseils qui m'aideront dans ma carrière de chercheur.

Je tiens également à remercier mes collègues du laboratoire Roberval, du département Génie des Systèmes Mécaniques et du Centre d'Innovation de l'UTC pour leur accueil et leur soutien pendant ces trois années. J'adresse également mes remerciements à mes collègues et amis doctorants et docteurs. "Parce qu'il y a une vie après la thèse" dixit le RED^2. Un remerciement particulier pour ceux avec qui j'ai débuté cette aventure : Fab, Chris, Marianne, Chen, Nico, Flo, Benji, Sarah, Pierre et Pierre, Justine, Andrea, Yorick et ceux qui nous ont rejoint par la suite : Cuong, Gaëtan, Fabien, Anaïs, Arnaud, Jonathan, Julia et Jinhua.

Je remercie ma famille et mes amis de longue date de m'avoir épaulé pendant ces trois années et d'avoir cru en moi. De chaleureux remerciements à toi, Antoine, pour ton soutien indéfectible depuis novembre 2013 ; la thèse en aura été que meilleure grâce à toi.

Mes derniers remerciements iront à mon père, un citoyen du monde qui m'a transmis la passion de la mécanique depuis mon plus jeune âge mais aussi le goût du voyage, l'amour des autres, la persévérance et bien d'autres valeurs avant de nous quitter en mars 2014.

Mardi 22 mars 2016

Table des matières

Table des figures

Liste des tableaux

Abréviations

BDC	Base de Connaissances
CAO	Conception Assistée par Ordinateur
HDI-RE	Heterogeneous Data Integration for Reverse Engineering
KBE	Knowledge-Based Engineering
NURBS	Non-Uniform Rational Basis Splines
RE	Reverse Engineering

Introduction générale

"On mesure l'intelligence d'un individu à la quantité d'incertitudes qu'il est capable de supporter."

- Emmanuel Kant

Depuis ces dix dernières années, les outils liés à l'ingénierie n'ont cessé d'évoluer. Grâce à l'accessibilité constante des solutions de conception assistée par ordinateur, le Reverse Engineering (RE) est devenu un outil pratique et abordable qui a su s'intégrer au sein des activités liées à l'ingénierie [Thilmany, 2012]. Dans nos travaux, nous l'avons défini comme un processus qui débute d'un produit physique et qui permet d'aboutir à une représentation numérique en deux ou trois dimensions. Il peut être utilisé à différents stades du cycle de vie du produit et de manière routinière. Que ce soit en bureau des méthodes, en fabrication ou en maintenance, différents types de données sont attachés à la maquette numérique [1] du produit tels que des manuels de maintenance, des Avant-Projet d'Études de Fabrication, des photographies, etc. Ces données restent souvent en marge car leur utilisation est restreinte à un besoin spécifique (manuel de maintenance pour une opération de maintenance) alors qu'elles pourraient être utiles dans le cadre du RE .

De plus, la plupart des outils de RE proposent de reconstruire le modèle CAO avec une géométrie morte, du moins lorsque cela est fait de manière automatique. Récupérer les connaissances métiers en même temps que le modèle numérique auquel elles appartiennent, est un des enjeux actuels.

Pour chaque étape du processus de RE, on peut constater des carences notamment en termes d'automatisation des processus ou de quantité et de complétude des données utilisées. En effet, scanner un assemblage de plusieurs composants (étape d'acquisition) est coûteux en temps d'autant plus lorsqu'il est question de scanner séparément chaque pièce. Une solution permettant de gérer les assemblages mécaniques permettrait un gain de temps considérable tout en restant peu onéreuse, ce qui serait un plus.

Ensuite, le traitement des données issues du scan (étape de traitement) est un travail très fastidieux surtout lorsqu'on souhaite obtenir un modèle CAO qui soit propre, paramétrable et réutilisable dans le processus d'ingénierie. L'intégration de données hétérogènes [2] permettrait un gain de temps en enrichissant les informations nécessaires à la reconstruction du modèle 3D.

Enfin, en fonction du niveau de détails [3] souhaité en sortie (étape de reconstruction de la maquette numérique), les composants de l'assemblage scanné pourraient être identifiés automatiquement. Ainsi, un modèle paramétré de chaque pièce serait instancié et servirait à reconstruire partiellement ou intégralement [4] la maquette numérique du produit.

1. Maquette numérique : ensemble de données numériques permettant de représenter un objet ou un ensemble de pièces constituant un assemblage. Ces données sont de différentes natures : solide 3D (pour la représentation des pièces), squelette de structure ou arborescence du produit (pour l'organisation des données), notes de calcul (pour les connaissances métiers)...

2. Données hétérogènes : données de différentes sources appartenant au produit tels que les dessins de définition, les nuages de points, les modèles CAO, les photographies, etc.

3. Il s'agit du niveau d'informations fourni par la maquette numérique en sortie du processus de RE : nomenclature simple, nomenclature + modèles CAO paramétrés etc.

4. Selon le contexte de RE et le besoin de l'opérateur, la maquette peut être mise à jour au niveau d'un sous-assemblage ou elle peut impacter l'assemblage au complet.

Résumé de la situation idéale : nous recherchons une méthodologie de Reverse Engineering permettant l'intégration de données hétérogènes afin de récupérer des modèles CAO paramétrés pour des assemblages mécaniques.

Verrous scientifiques

Le premier verrou scientifique repose principalement sur l'inexistence de solutions qui permettent le traitement des données hétérogènes acquises par scan ou autre et en particulier concernant les assemblages.

Un deuxième verrou concerne la complétude des données (surfaces de contact entre les pièces non numérisées) ainsi que la distinction des limites entre les composants d'un assemblage numérisé. Le faible niveau d'information interfère avec le souhait de reconstruire un maquette numérique avec des modèles CAO paramétrés.

Le dernier verrou concerne la comparaison des données. Le moyen de comparaison doit pouvoir s'adapter à l'existence de données de différents types. De plus, une capitalisation de ces connaissances doit être possible afin d'accélérer la reconstruction de la maquette numérique, dans le cas de RE routinier [5].

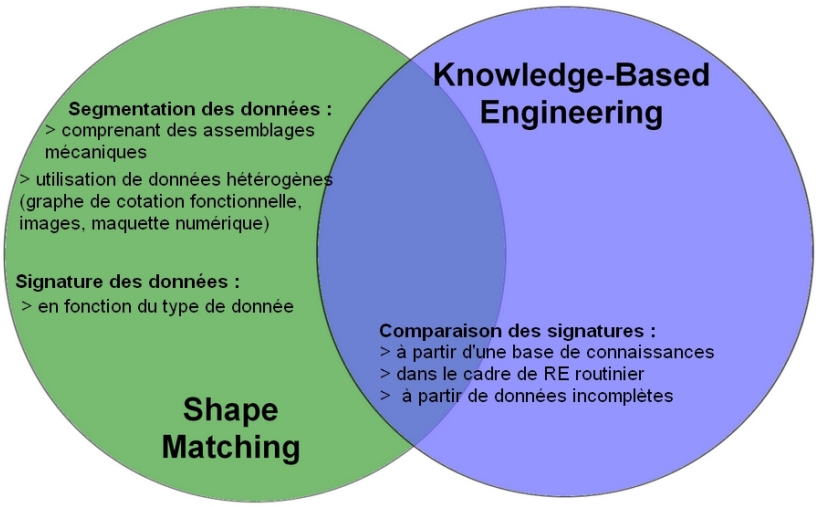


Figure 1 – Domaines de recherche de ces travaux.

La figure 1 résume les différents points abordés dans cette étude par rapport aux domaines de recherche.

Proposition

Nous proposons une méthode permettant **l'intégration** [6] **de données hétérogènes**. Le but sera de l'appliquer à des assemblages afin de récupérer la maquette numérique de **grands ensembles mécaniques de plusieurs centaines de pièces** (moteur de voiture, avion, sous-marin etc.). Nos travaux de recherche s'inspireront du domaine du "Shape Matching" (reconnaissance de forme) afin de traiter les données, notamment en ce qui concerne leur segmentation puis l'identification de formes caractéristiques.

Les informations métiers et la manière de les extraire des données d'entrée sont ce que nous appellerons la **signature** de notre donnée (acquise par photographie, scan ou autre). Cette signature apportera un niveau d'information relatif à la donnée signée et qui permettra, selon les cas, de palier au problème de complétude des données.

Enfin l'existence d'une BDC propre au domaine d'utilisation de notre processus de RE aidera à la reconstruction d'un **modèle CAO paramétré**. Le domaine du Knowledge-Based Engineering (KBE)

5. Nous considérons que l'activité de RE est répétitive et se concentre sur des assemblages dont les composants sont présents en base de connaissances (BDC). La fréquence est liée à l'activité de l'entreprise.

6. Intégration : nous le définirons par l'action d'extraire des informations des données hétérogènes grâce un processus de signature.

traite de la gestion des connaissances, leur traçabilité ainsi nos travaux s'appuieront sur les résultats dans ce domaine.

Notre proposition, nommée HDI-RE, repose sur une méthode se décomposant en trois étapes :

— la segmentation des données hétérogènes ;
— la signature permettant l'extraction des caractéristiques topologiques et géométriques ;
— la comparaison des signatures permettant d'identifier les composants grâce à une base de connaissances.

L'ensemble des solutions développées dans notre méthodologie seront ensuite utilisées dans une solution logicielle appelée "METIS" (**M**od**E**lisation **T**ridimensionnelle des maquettes numériques par l'**I**ntégration de données géométriques et de connaissances hétérogène**S**) et développée par la société DeltaCAD, éditrice de logiciels dans le domaine de la CAO. Cet industriel est aussi le porteur de projet sur lequel reposent ces travaux de thèse. Une bourse de l'**A**gence **N**ationale de la **R**echerche a permis de les financer (référence du projet : ANR-12-MONU-0004).

Structuration du manuscrit

Ce manuscrit se décompose en cinq chapitres : contexte et positionnement scientifique, état de l'art, proposition, expérimentation et validation puis conclusion et perspectives.

Le **premier** chapitre présentera le contexte industriel dans lequel s'insère notre méthodologie de Reverse Engineering ainsi que les besoins actuels de l'industrie. Une veille technologique sur les logiciels de RE sera également présentée.

Le chapitre **deux** permettra d'établir un état de l'art sur la segmentation des données (2D et 3D), leur signature et la comparaison des signatures dans un contexte de KBE.

Le chapitre **trois** exposera notre proposition pour répondre à la problématique et abordera les concepts sur lesquels s'appuie notre méthodologie HDI-RE. La notion de signature sera définie et les solutions choisies seront présentées.

Le chapitre **quatre** permettra d'illustrer nos travaux par un cas industriel. Les résultats des algorithmes développés seront présentés et analysés.

Le **dernier** chapitre comportera les conclusions ainsi que les perspectives envisageables à court et moyen termes.

CHAPITRE 1

Positionnement scientifique de la recherche : les nouveaux besoins en Reverse Engineering

"Un amour, une carrière, une révolution : autant d'entreprises que l'on commence en ignorant leur issue."

- Jean-Paul Sartre

Introduction

Ce chapitre consiste à définir le cadre de cette étude. Il est composé des sections suivantes :

— section 1.1 : le contexte de l'étude où nous présenterons une définition du RE et des étapes de son processus.
— section 1.2 : les raisons industrielles du RE dans lesquelles se positionnent nos travaux.
— section 1.3 : nous présenterons ici une veille technologique des solutions logicielles dans le RE ;
— section 1.4 : nous ferons une analyse des principales études dans le RE ;
— section 1.5 : une synthèse du positionnement de ces travaux sera faite et découlera sur l'énoncé de notre problématique et des hypothèses délimitant le cadre de l'étude. Le diagramme illustrant la méthodologie HDI-RE sera présenté et servira de fil rouge tout au long de ce manuscrit.

Nous avons choisi de présenter dans ce premier chapitre les principaux travaux dans le domaine du RE ainsi que les solutions logicielles existantes afin de traiter uniquement des travaux relatifs à notre proposition dans le chapitre d'état de l'art.

1.1 Le contexte de l'étude

Selon le domaine et l'application, différentes traductions existent pour le terme "reverse engineering" tels que "rétro-conception", "rétro-ingénierie" ou encore "décompilation" dans le domaine de l'informatique. Cependant ce terme anglais tend à entrer dans la langue française et nous choisirons de garder la dénomination "Reverse Engineering" (RE) tout au long de ce manuscrit de thèse. La figure 1.1 de [Raja et Fernandes, 2007] illustre le positionnement du "Reverse Engineering" par rapport aux autres activités d'ingénierie. De la gauche vers la droite, on retrouve les outils de CAO (*CAD* en anglais), FAO (*CAM* en anglais) et de prototypage rapide (*RP* en anglais). Le RE permet alors un retour du produit fini vers le modèle CAO (double flèche).

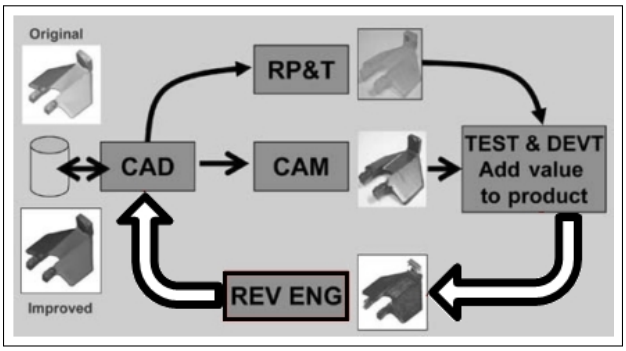


Figure 1.1 – Le RE par rapport aux activités d'ingénierie [Raja et Fernandes, 2007].

Dans la littérature, suivant le domaine d'application plusieurs définitions du Reverse Engineering existent comme :

- le processus d'extraction des données de conception et fabrication d'une pièce existante [Motavalli, 1998] ;
- le concept permettant de fabriquer une pièce à partir de son modèle physique sans utiliser aucun dessin de définition [Abella *et al.*, 1994] ;
- le processus permettant de récupérer la nouvelle géométrie d'une pièce fabriquée en la numérisant et en modifiant son modèle CAO existant [Yau *et al.*, 1993]

et les applications en mécanique sont diverses [Raja et Fernandes, 2007] :

- inspecter et comparer la géométrie de l'objet réel avec le modèle CAO ;
- mesurer le comportement d'outils de fabrication (contrôle de l'usure par exemple) ;
- capturer la géométrie d'usine ou de chaîne de production afin d'utiliser des logiciels CAO pour la conception de canalisations, chaufferie, ventilations, climatisation ;
- capturer les déformations d'un objet, dans un cas de charge d'une poutre par exemple ; afin de comparer la pièce déformée avec les prévisions de l'analyse des éléments finis ;
- capturer les formes d'une pièce existante dont le modèle CAO n'existe pas afin qu'une pièce de rechange soit fabriquée par prototypage rapide par exemple.

On retrouve certaines de ces applications dans d'autres domaines comme en médecine afin de concevoir et fabriquer des prothèses [Colombo *et al.*, 2010], en muséologie [Laroche *et al.*, 2008], en architecture [De Luca, 2006], en criminologie afin d'inspecter les scènes de crimes plus précisément [FaroLaser, 2012] etc.

Dans la majorité des études, le processus de RE se décompose en trois grandes étapes :

1. l'acquisition des données : l'objet physique est numérisé en données de deux ou trois dimensions ;
2. le traitement des données : les données peuvent être segmentées par exemple afin d'extraire des surfaces canoniques (plan, cylindre, sphère) ou des surfaces complexes ;
3. la reconstruction du modèle CAO : les surfaces extraites peuvent par exemple être épaissies et si nécessaires retravaillées afin de reconstruire un solide en 3D.

Selon les moyens utilisés (matériel d'acquisition, solutions informatiques), le processus est plus ou moins automatisé, certaines étapes étant réalisées manuellement par l'opérateur. De plus, ces travaux traitent d'assemblages mécaniques de plus ou moins grande taille (de plusieurs dizaines de composants), les opérations qui se raportent au RE sont alors très chronophages et fastidieuses. D'où l'intérêt de porter notre attention sur les solutions aidant les utilisateurs aussi appelées "Computer-Aided Reverse Engineering" (CARE) [Raja et Fernandes, 2007, chapitre 2] qui sont les outils équivalents à ceux communément appelés CAD/M/E (pour Computer-Aided Design / Manufacturing / Engineering) pour le RE. La figure 1.2 propose une corrélation entre les outils de conception (CAD) et de rétro-conception (CARE).

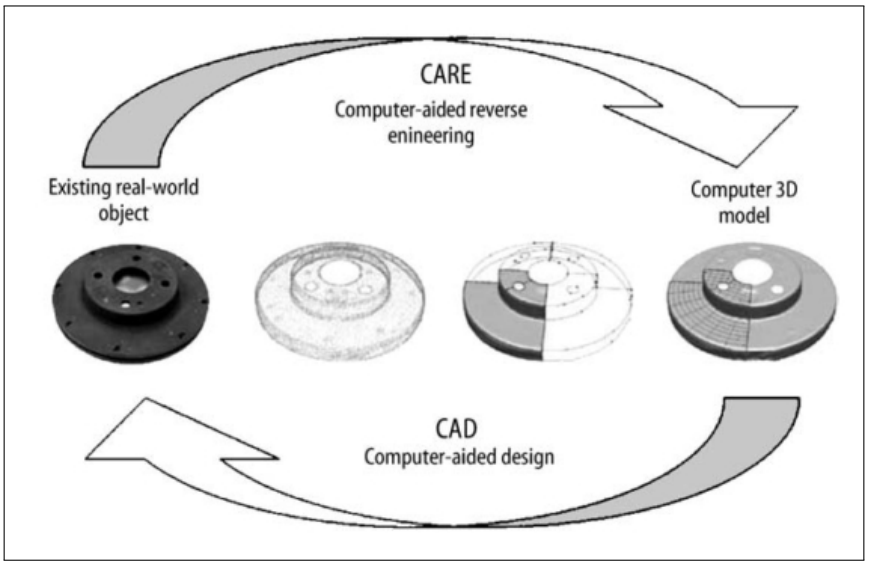


Figure 1.2 – Reverse Engineering assisté par ordinateur (Computer-Aided Reverse Engineering) comparé à la conception assistée par ordinateur (CAO) d'après [Raja et Fernandes, 2007].

Dans le processus du RE et en particulier lors de l'**étape d'acquisition** des données, plusieurs types de technologies existent. Dans ces travaux, nous considèrerons l'utilisation de matériel standard pour l'acquisition des données 2D tels que les appareils photo qui fournissent des photographies sous le format .bmp ou .jpeg. Cependant nous nous intéresserons en priorité aux données 3D ; ce type de données apportant le plus haut niveau d'information (géométrie et topologie).

Il existe alors deux technologies d'acquisition pour les données 3D : avec contact et sans contact. Nous nous concentrerons en particulier sur les deuxièmes qui permettent de scanner des pièces de grandes dimensions et notamment des assemblages. En sortie de cette étape d'acquisition, nous obtiendrons un nuage de points dont les points sont définis dans l'espace (x, y, z). Le format utilisé majoritairement est le STL (comme stéréo-lithographie). C'est dans ce contexte d'acquisition que se placent nos travaux.

Pour ce qui est de l'**étape de traitement des données**, il est important de définir son but. Il s'agit d'extraire des informations pertinentes telles que les informations topologiques ou géométriques mais aussi celles permettant de caractériser la donnée (critère statistique par exemple). Elles permettent ensuite de reconstruire la maquette numérique. Ensuite, l'obtention des données dans le cas d'assemblages de plusieurs pièces fournit un ensemble de points ou de pixels pour lesquels, à l'état brut, il est impossible de déterminer les limites de chaque composant de manière spontanée. Dans la majorité des cas, on effectue alors une segmentation de cette donnée. En fonction de la nature de la donnée (2D ou 3D),

la segmentation est un processus de découpage de la donnée en segments (ensemble de pixels ou de triangles d'un maillage issu d'un nuage de points). Ce découpage permet de caractériser chaque segment de la donnée initiale en vue de l'identifier. Ainsi il sera possible d'identifier partiellement un composant à l'intérieur d'un assemblage.

Une étude approfondie sera menée dans le chapitre 2 concernant la segmentation pour chaque type de donnée (2D et 3D). Suivant le cas, les segments extraits aussi appelés régions sont ensuite analysés afin d'en déterminer des caractéristiques (topologie pour la 3D ou contours pour une image). La formalisation de ces caractéristiques est ce que nous appellerons "signature" de la donnée. C'est le cœur des travaux présentés dans ce manuscrit.

Le terme "signature" est alors défini par un "ensemble de caractéristiques permettant de décrire une donnée". Il vient alors l'étape de comparaison des signatures les unes par rapport aux autres et qui permet ensuite d'identifier le segment voire la donnée dans sa globalité. C'est à cette étape du processus que nous ferons appel aux travaux dans le domaine de la reconnaissance de forme (aussi appelé *Shape Matching*) ou encore de la vision par ordinateur [1] (connue sous le nom de *Computer Vision*).

Les travaux dans le RE sont étayés par ceux du domaine du *Shape Matching* [Veltkamp, 2001] car ils permettent d'identifier des formes puis des composants dans le nuage de points et cela grâce à la mesure de la similarité (aussi appelée étape de comparaison).

Pour la dernière étape du processus de RE à savoir la **reconstruction du modèle** CAO, il s'agit généralement d'utiliser les informations extraites de l'étape précédente, afin de reconstruire la géométrie mais aussi la topologie de chaque composant de la maquette. Dans le cadre de cette étude, nous nous intéresserons plus globalement à la reconstruction de la maquette numérique au delà du modèle CAO. Ce dernier est une représentation de la maquette mais pas l'unique. En sortie du processus de RE, il est possible d'obtenir différents niveaux d'information ; cela va de la simple nomenclature au modèle CAO paramétré.

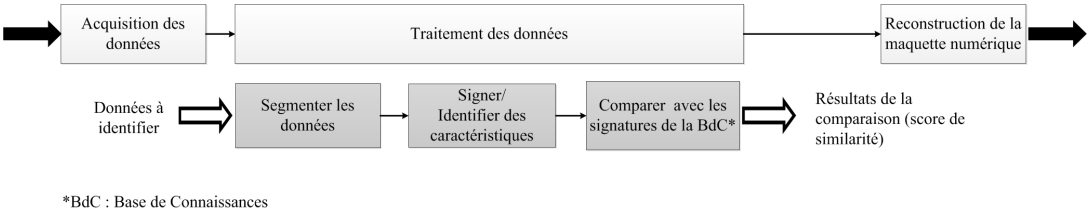


Figure 1.3 – Les étapes de notre méthodologie par rapport à la démarche globale de RE (au-dessus).

Afin de clarifier cette section, la figure 1.3 résume les étapes sur lesquelles se positionnent nos travaux (cases grisées) par rapport à la démarche globale de RE (au-dessus).

1.2 Les raisons industrielles du Reverse Engineering

A l'ère du tout-numérique, de nombreux industriels (DCNS ou Dassault Aviation) ont réussi à faire valider leur produit par le bureau VERITAS^{TM} en se dispensant de prototype coûteux et non-commercialisable [Picard, 2009]. La maquette numérique occupe donc une place importante ; mais encore faut-il que ces données virtuelles soient identiques au modèle physique tout au long de la vie du produit. D'où cette nécessité de maintenir à jour cette maquette que ce soit au fur et à mesure de la fabrication du produit, comme tout au long de sa vie. Peu de solutions logicielles sur le marché répondent à cette problématique et encore moins lorsqu'il s'agit de grands ensembles mécaniques.

La maquette numérique évolue, en effet, tout au long du cycle de vie du produit tel que nous pouvons l'observer dans la figure 1.4. A chaque étape du cycle de vie, le produit est sous différentes formes : physique ou numérique (2D ou 3D). Attaché à la maquette numérique, on retrouve également un ensemble de documents, de données dites hétérogènes qui dans notre cas, seront utilisées afin d'enrichir les données issues de scan par exemple. Grâce aux travaux de [Durupt, 2010; Rathore et Jain, 2014; Wang, 2013]

1. Vision par ordinateur (Computer Vision) : c'est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs données 2D ou 3D prises par un système d'acquisition (par exemple une caméra).

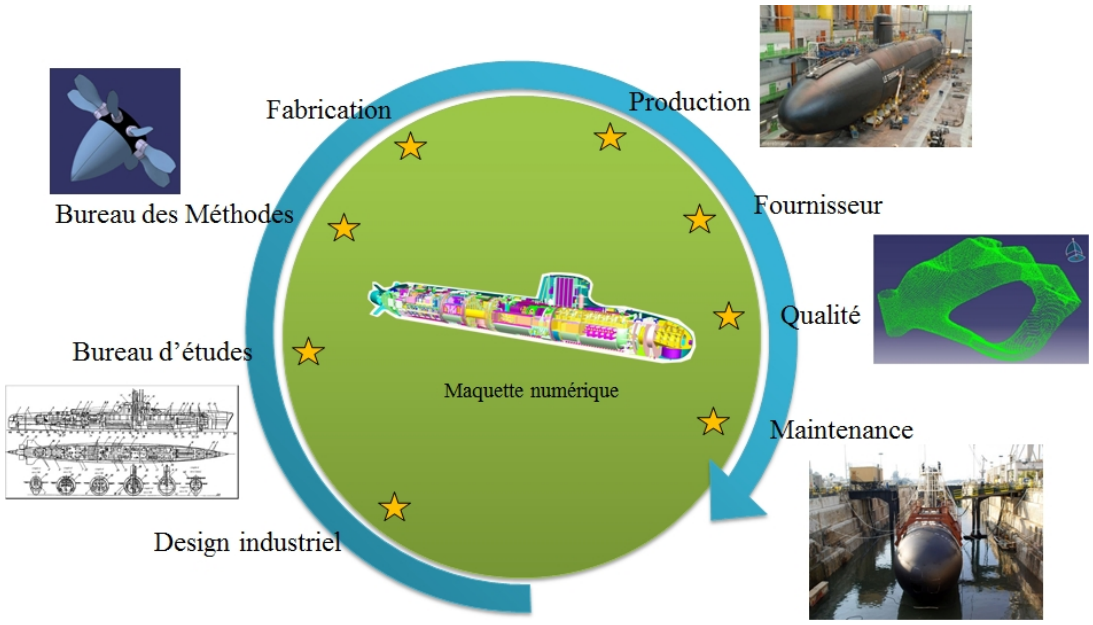


Figure 1.4 – Les étapes du cycle de vie d'un produit dont certaines pouvant intégrer l'activité de RE.

et au contexte industriel sur lequel repose ces travaux, nous avons identifié trois cas d'étude permettant d'identifier les raisons industrielles qui amènent à recourir au RE :

1. en bureau d'études lors de phases de l'analyse de l'existant ou lorsque toute trace numérique du produit a été perdue ("*Reverse Engineering from scratch*") : c'est le cas le plus connu puisque c'est une des motivations principales des industriels dans l'utilisation des technologies de RE. Ce cas industriel est le plus complexe car on espère récupérer le modèle CAO natif [2] sans posséder aucune donnée (2D ou 3D) au préalable ; permettant ainsi de ré-industrialiser le produit à l'identique. Ainsi les temps associés aux phases de conception, calculs et simulation et par conséquent le délai de mise sur le marché du produit sont réduits considérablement. Une autre raison pour le "RE from scratch" concerne la numérisation de pièces ayant une longue durée de vie (supérieure à 40 ans). Dans ce cas industriel, nous souhaitons récupérer en sortir du processus de RE une maquette numérique de notre assemblage mécanique comprenant des modèles paramétrés afin de pouvoir industrialiser le produit. C'est le cas dans la figure 1.5 où une pièce doit être remplacée sur un véhicule ancien dont on ne dispose aucune donnée de conception ou autre.
2. en production ("*As designed / as made*") : le produit fabriqué ou en cours de fabrication est numérisé puis nous comparons cette donnée 3D (ou 2D) avec le produit numérique initial telle qu'une maquette numérique afin de s'assurer de sa conformité (mise à jour de sa nomenclature, contrôle du montage, inspection qualité, etc.). L'industrie fait appel au RE dans le cas de très grands ensembles mécaniques comme les avions, les bateaux ou les sous-marins afin d'automatiser ces opérations de contrôle routinières et ainsi assurer le maintien de la maquette numérique à jour (voir figure 1.6). A ce stade, il ne sera pas nécessaire de récupérer les modèles CAO en sortie de notre processus de RE. En effet, la comparaison pouvant s'effectuer dans un premier temps au niveau des nomenclatures, puis si nécessaire, avec les modèles CAO reconstruits à partir de l'assemblage scanné pour une comparaison géométrique par exemple.
3. en maintenance ("*As designed / as maintained*") : on compare le produit de la même manière que pour le deuxième et on s'assure que les données CAO ne sont pas obsolètes. En effet, pour les produits mécaniques ayant une durée de vie assez longue, 20 à 30 ans pour un avion par exemple, de nombreux composants sont changés au cours des différentes opérations de maintenance. Un constructeur avionique peut perdurer dans le temps mais ce n'est pas forcément le cas de ses sous-traitants qui peuvent être amenés à disparaître ou à changer. De nouvelles pièces peu ou presque

2. Il s'agit d'un modèle CAO dont la géométrie est modifiable et/ou paramétrable (ce qui n'est pas le cas d'un solide mort).

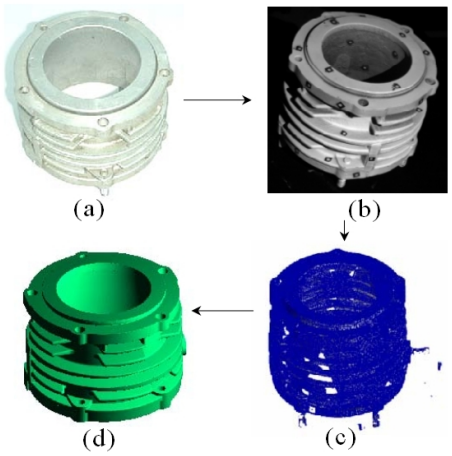


Figure 1.5 – *Reverse engineering from scratch* d'un séparateur de filtre en aluminium pour véhicule de l'armée indienne : (a) pièce originale, (b) pièce originale avec des marqueurs, (c) nuage de points et (d) modèle CAO [Pal *et al.*, 2006].

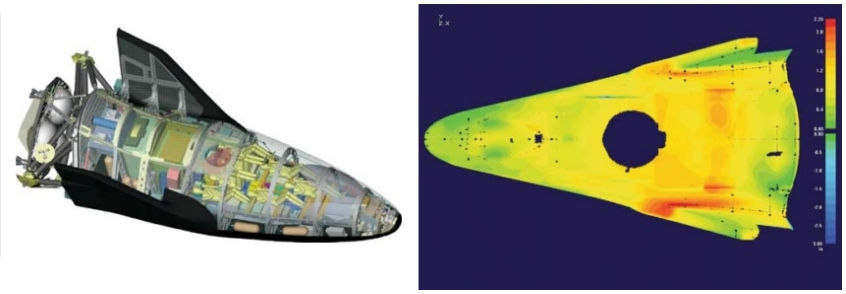


Figure 1.6 – Reverse engineering (*As designed / as made*) d'un engin spatial afin de visualiser les écarts dimensionnels entre l'assemblage scanné et sa maquette CAO de référence (Copyright©Aerospace Manufacturing Technologies, Inc., Arlington, WA, USA d'après [Raja et Fernandes, 2007]).

identiques sont alors adaptées sur l'avion lors de sa maintenance. Il est alors essentiel de propager ces modifications jusqu'à sa maquette numérique afin que cette dernière soit à jour. Le RE permet ainsi d'effectuer ce retour d'informations du produit physique jusqu'à sa donnée numérique initiale. Tout comme pour le deuxième cas industriel, la comparaison du produit peut s'effectuer au niveau des nomenclatures. Une autre application du RE en maintenance consiste à comparer le produit physique avec son modèle CAO afin d'observer son usure et prévoir sa maintenance : réparation par ajout de matière (fabrication additive) comme par exemple dans l'exemple de la figure 1.7. On retrouve également beaucoup de travaux traitant de réparation ou remplacement de pièces endommagées grâce au RE comme dans les travaux de [Dúbravčík et Štefan Kender, 2012] et [Bagci, 2009].

Après avoir présenté les différents cas d'application possibles de l'activité de RE tout au long du cycle de vie d'un produit, nous allons maintenant nous intéresser aux différents travaux dans ce domaine et en particulier les études scientifiques ainsi que les logiciels existants sur le marché.

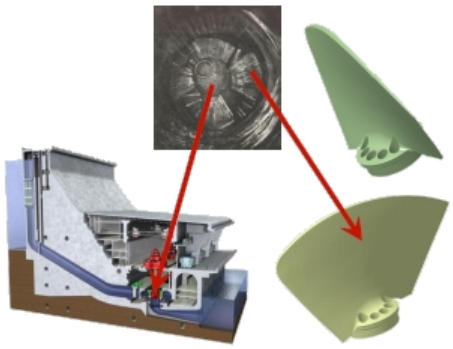


Figure 1.7 – Reverse engineering (*As designed / as maintained*) de pales usées d'une turbine de barrage hydro-électrique pour rechargement en matière ou re-fabrication [Gélineau, 2013].

1.3 Veille technologique sur les logiciels de Reverse Engineering

Dans cette section, nous présenterons une sélection de différentes solutions logicielles présentes sur le marché qui traitent l'activité de RE en partant de données issues de scan jusqu'à la reconstruction du modèle 3D.

Geomagic Spark™ propose une méthode automatique de conversion des objets physiques en modèle numérique 3D à l'aide de nuage de points du modèle physique. L'outil propose différentes fonctionnalités comme la réparation des géométries endommagées grâce à la possibilité d'éditer les nuages de points et maillages ; la détection automatique des volumes permettant de reconstruire un modèle CAO et pour finir la création rapide de plans techniques 2D annotés et de cotes 3D. En ce qui concerne les assemblages, il est possible de générer une nomenclature avec des composants à part entière et ce à partir d'un même et seul nuage de points (scan d'un assemblage complet par exemple). Enfin, *Geomagic Spark* est compatible [3] avec de nombreux modeleurs CAO. Cependant le paramétrage des modèles CAO n'est pas possible étant donné que la méthodologie de rétro-conception proposée n'emploie pas les esquisses. Un aperçu de *Geomagic Spark*™ est présenté dans la figure 1.8.

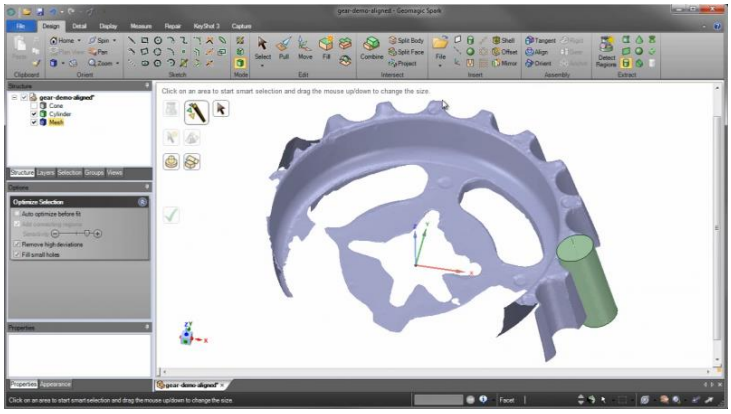


Figure 1.8 – Extraction de solide à partir de nuage de points partiel dans *Geomatic Spark*.

3DReshaper™ permet la reconstruction de modèles CAO à partir de maillages et nuages de points. La méthodologie s'appuie sur la création de polylignes qui permettent ensuite la génération de maillages puis de surfaces Non-Uniform Rational Basis Splines (NURBS) [4] qui sont ensuite épaissies, recréant ainsi

3. Les fichiers de sortie sont enregistrés dans un format propre à certains modeleurs, permettant leur édition.
4. NURBS (B-Spline) : "représentations mathématiques de la géométrie en 3D pouvant décrire avec précision toute forme, d'une simple ligne 2D, un cercle, un arc ou une courbe à une surface ou un solide organique 3D de forme libre très complexe" [Rhinoceros3D, 2012].

un modèle 3D numérique. Cependant aucun arbre de construction n'est créé lors de la re-conception et aucune entité géométrique n'est identifiée (cylindre, plan, etc.). Le modèle CAO obtenu est un solide avec une géométrie morte. La figure 1.9 présente les étapes de reconstruction du modèle CAO dans *3DReshaper*™.

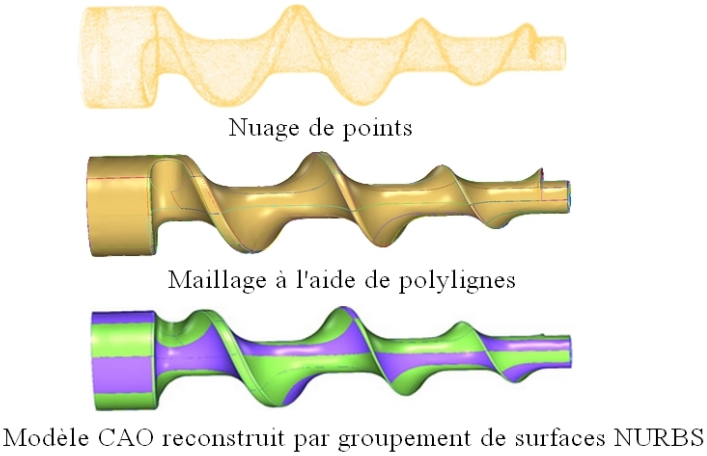


Figure 1.9 – Du nuage de points à la CAO dans le logiciel *3DReshaper*.

Pro Engineer Reverse Engineering(REX)™ s'appuie aussi sur la génération de courbes sur une facette, une surface, à partir d'une limite de surface, via des points. Des surfaces sont ensuite obtenues permettant la création du modèle CAO équivalent, le tout en corrigeant les défauts dans les données issues du scan (rognage de points, réduction du bruit, suppression de points excentrés, etc.). De même qu'avec *3DReshaper*, aucune nomenclature est créée lors de la re-conception et aucune entité géométrique n'est identifiée (cylindre, plan, etc.).

RapidForm XO Redesign™ (maintenant appelé *Geomagic Design X* ®- fusion des deux logiciels) est un outil de génération automatique de modèles CAO à partir de données numériques issues de scan 3D. Dans les modules proposés par ce logiciel, l'un propose une segmentation automatique du nuage ainsi qu'une reconnaissance du type (cylindre, sphère, etc.) de chaque région. Les esquisses du solide initial sont générées de manière automatique par sélection de surfaces par l'utilisateur puis éditées si nécessaire et enfin épaissies. L'objet est alors reconstruit "volume après volume" tel que nous pouvons l'observer dans la figure 1.10. La modélisation finale peut ensuite être enregistrée sous différents formats (compatibles avec les modeleurs CAO tels que *SolidWorks*, *CATIA* ou *ProEngineer*). Une comparaison des logiciels *RapidForm* et *Geomagic* a été réalisée par [Chang, 2011] du temps où ils étaient encore deux logiciels distincts. A cette époque, seul le logiciel *RapidForm* permettait de reconstruire des modèles CAO paramétrés. Il possédait également les mêmes fonctionnalités que les modeleurs CAO notamment en ce qui concerne la création d'un arbre de construction ou encore l'édition des fonctions.

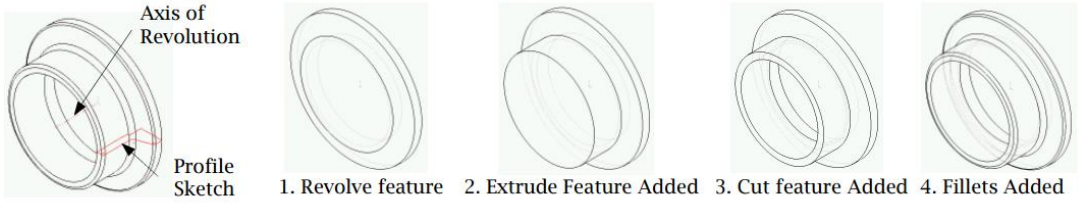


Figure 1.10 – Reconnaissance des caractéristiques de la pièce dans le logiciel *Geomagic*.

Le module *Quick Surface Reconstruction* (QSR) du logiciel *CATIA*^{TM} permet de reconstruire des surfaces géométriques à partir de nuages de points de manière semi-automatique ou automatique. Il crée automatiquement des courbes frontières par reconnaissance de la géométrie numérisée. Il s'en suit la création automatique des surfaces par reconnaissance des formes canoniques (plan, sphère, cylindre, cône), présentes dans le nuage de points. Les surfaces obtenues sont ensuite "remplies" en fonction des conditions de tangence et courbure spécifiées, permettant de retrouver un modèle CAO exploitable dans tout le logiciel *CATIA*.

Le logiciel *PolyWorks*^{TM} propose un outil d'extraction des géométries et des courbes à partir de maillages polygonaux. Ces derniers peuvent être transformés en surfaces gauches ou en entités standards. Pour cela des surfaces NURBS sont alors ajustées sur le maillage comme des "patches". Ces surfaces peuvent être ensuite exportées en tant que modèle CAO.

Le logiciel *Rhinoceros 3D* ^{TM} connu initialement dans le domaine du design est désormais utilisé dans le RE. Des polylignes sont créées à partir de maillages. Plusieurs outils d'édition de courbes, de création et d'édition de surfaces et de solides sont disponibles. Il propose notamment un plugin Rhinoterrain^{TM} permettant de rétro-concevoir des bâtiments ou des paysages dans les moindres détails à partir de nuages de points ou de données photogrammétriques et topographiques. Les données en sortie permettent surtout de créer des rendus proches de la réalité mais ce sont des solides morts (ou seulement modifiables dans ce logiciel).

1.4 Synthèse des principaux travaux dans le RE

Dans cette section, nous présenterons les principales études scientifiques dans le domaine du RE dont le positionnement scientifique est proche du nôtre. Nous les classerons par ordre chronologique afin de mesurer l'évolution des travaux dans le domaine. Nombreux de ces travaux appartiennent au domaine du bâtiment et traitent du *Building Information Modeling* (BIM). Cette activité est très proche de celle du RE puisque les trois étapes fondamentales sont l'acquisition des données, la segmentation et enfin la modélisation. Les récents travaux de [Hichri *et al.*, 2013] proposent un état de l'art dans le BIM. Nous présenterons ainsi certains des travaux appartenant à ce domaine que nous jugerons utiles. Cependant il est important de rappeler que nos travaux traitent de RE appliqué à l'industrie manufacturière.

Parmi les plus anciens travaux, nous pouvons citer ceux de [Várady *et al.*, 1997] qui proposent un état de l'art général de l'activité de RE et en particulier pour les étapes d'acquisition des données, de segmentation, d'ajustement de surface et pour la reconstruction du modèle B-Rep (*Boundary Representation*). Cette étude permet de mettre en lumière les principales difficultés rencontrées lors de l'activité de RE et les perspectives d'études autour de cette dernière.

Ensuite nous pouvons citer [Benko *et al.*, 2001] qui proposent une méthode de RE en onze étapes :

1. Acquisition des données
2. Fusion des points confondus
3. Maillage/Décimation du nuage
4. Segmentation
5. Reconnaissance des surfaces simples
6. Reconstruction des extrusions linéaires et des surfaces de révolution
7. Reconstruction des surfaces lisses
8. Construction d'un graphe d'adjacence
9. Identification des contraintes
10. Création du modèle B-Rep
11. Ajout des arrondis

Nous intéresserons aux étapes 8 et 9. Le graphe d'adjacence reflète la topologie du B-Rep final. La continuité entre les triangles et les contraintes est vérifiée : par exemple, si le sommet d'une région issue de la segmentation est partagé avec au moins trois régions et s'il est situé à l'extrémité des polylignes, alors il est considéré comme un sommet du modèle final.

Les travaux de [Lartigue *et al.*, 2002] offrent une étude concernant la qualité des nuages de points lors d'activités de Reverse Engineering ou d'inspection de pièces. Pour cela des indicateurs sont proposés afin de permettre à l'utilisateur d'évaluer la qualité de ses données. Parmi les problèmes qui entravent la bonne qualité de la donnée, ils citent les données bruitées et non-homogènes, les nuages de points incomplets (avec des trous) ou encore des nuages imprécis. Le but de l'évaluation proposée est de garantir une concordance entre les données brutes et les intentions de l'utilisateur (re-conception du modèle 3D, inspection ...). Les travaux récents de [Quinsat et Lartigue, 2015] proposent une solution pour pallier au problème d'incomplétude des données et en particulier en ce qui concerne les nuages de points avec des trous. La méthode proposée repose sur la comparaison entre le modèle numérique (donnée antérieure au processus de RE par exemple) appelé maillage nominal, et le maillage issu du nuage de points à réparer (avec des trous). Une illustration des résultats obtenus grâce à leur méthodologie est proposée dans la figure 1.11.

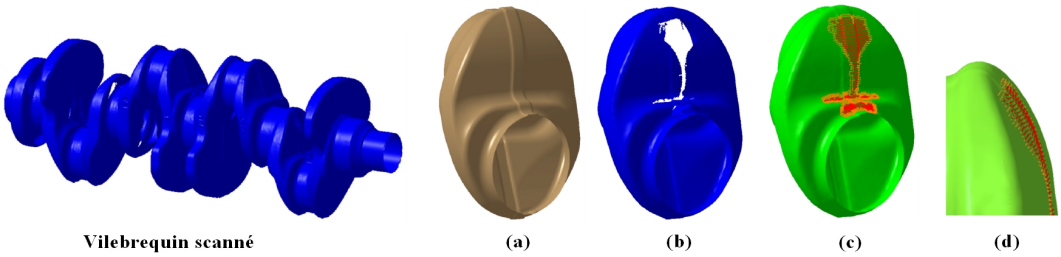


FIGURE 1.11 – Remplissage de trous pour un vilebrequin scanné : (a) le maillage nominal (issu du modèle CAO, (b) la pièce numérisée, (c) transformation du maillage et (d) détails du trou rebouché d'après [Quinsat et Lartigue, 2015].

Le projet VPERI (Virtual Parts Engineering Research Initiative) a été lancé par [Army Research Office, 2003] (États-Unis) afin de fournir la vision, la stratégie et une méthodologie pour solutionner des problèmes de maintenance de produits à longue durée de vie. Une interface utilisateur nommée ASUDAL a été développée afin de permettre l'ajout de connaissances sous la forme d'équations algébriques correspondant aux fonctions de comportements des composants, aux lois physiques qui régissent leur comportement, leur disposition spatiale, etc. Cette interface fournit une aide aux concepteurs qui vérifient que le cahier des charges ait été correctement respecté et leur apporte un soutien pour explorer d'autres alternatives en les assistant lors des changements effectués.

[Deveau, 2006] propose une méthodologie de RE de façades de bâtiment à partir de deux types de données en entrée : les nuages de points et des images haute résolution, de densité supérieure à l'acquisition laser. Il examine l'intérêt de l'utilisation conjointe de nuages de points qui fournissent une information géométrique dense et des images qui apportent les informations géométriques et radiométriques. Les images l'aident à déterminer les limites entre les objets et les nuages de points pour récupérer la géométrie des surfaces. Il réussit ainsi à alléger les temps d'acquisition laser et de post-traitement. Comme l'illustre la figure 1.12, les données sont d'abord segmentées séparément puis les résultats sont fusionnés. Cette fusion est réalisée par mesure de ressemblance ou de dissemblance entre les éléments (régions par exemple).

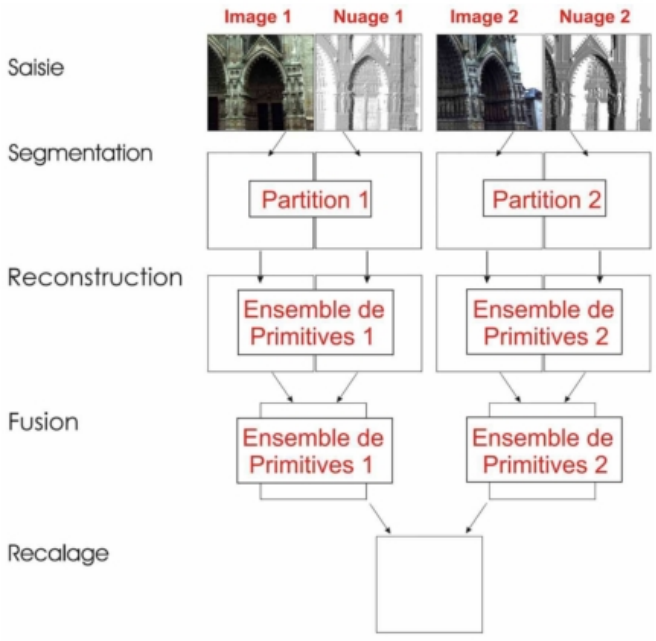


Figure 1.12 – Stratégie de RE proposée par [Deveau, 2006] combinant images et nuages de points.

Les travaux de [Wang *et al.*, 2012] proposent une méthodologie capable d'extraire automatiquement les caractéristiques géométriques à partir de nuages de points bruités et de reconstruire un modèle CAO paramétré, contenant les entités métiers. La figure 1.13 présente les différentes étapes. On retrouve en sortie un modèle CAO avec un arbre de construction qui traduit les intentions de conception. En plus de cela, les fonctions (extrusion, poche, etc.) possèdent des relations parents/enfants permettant de propager les modifications lors de l'édition de la pièce ; facilitant la réutilisation du modèle CAO.

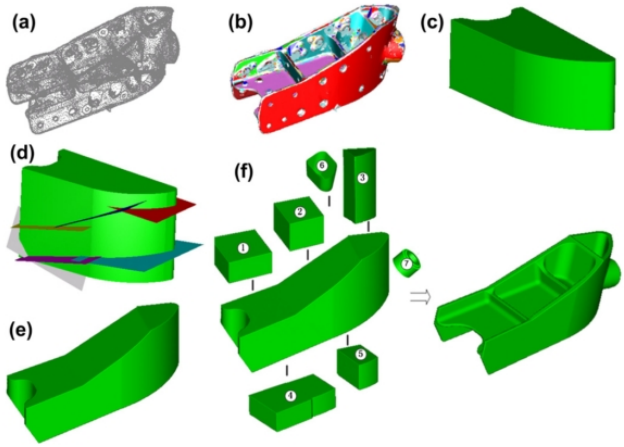


Figure 1.13 – Reconstruction d'une pièce d'avion [Wang *et al.*, 2012] : (a) le nuage de point initial ; (b) le résultat de la segmentation ; (c) extrusion d'un bloc à partir des surfaces extérieures ; (d) plans ajustés aux points correspondants ; (e) résultat intermédiaire en recoupant le bloc initial avec les plans ; (f) le modèle final reconstruit, généré en réalisant des opérations booléennes et des découpages de surfaces sur le volume intermédiaire.

[Herlem *et al.*, 2012] proposent une méthodologie de RE afin de comparer la maturité des maquettes numériques dans un contexte de gestion des changements (suppression, remplacement, déplacement de composants). Il compare le produit réel avec la maquette initiale puis il la met à jour afin d'améliorer le cycle de vie du produit. Il utilise la méthode par graphe de Reeb comme signature topologique (voir figure 1.14) ainsi que les informations de cinématique entre les composants du mécanisme de l'assemblage mécanique qui lui permettent de décrire l'assemblage du point de vue des connaissances métiers.

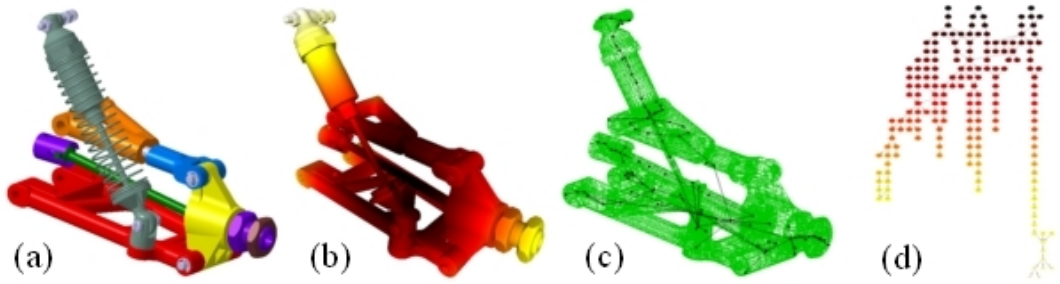


Figure 1.14 – Application des graphes de Reeb sur une suspension automobile [Herlem *et al.*, 2012] : (a) modèle CAO, (b) scan de l'assemblage physique, (c) affichage 3D du graphe de Reeb et (d) graphe de Reeb dans une seule dimension.

Dans son mémoire de thèse, [Bénière, 2012, chapitre 6] met en avant une méthode automatique de RE à partir de maillage 3D. Elle est décomposée en quatre étapes : 1°) l'extraction des primitives, 2°) la détermination du graphe d'adjacence, 3°) la construction des contours des faces et 4°) la construction du modèle B-Rep. Le graphe d'adjacence permet de définir les relations de voisinage entre les primitives extraites dans l'étape 2. Suivant sa morphologie (graphe cyclique par exemple), il est possible de déterminer les points d'intersection entre les primitives puis dans un deuxième temps les contours des faces.

Pour finir, nous aborderons les travaux récents de [Rathore et Jain, 2014] qui présentent une synthèse des études dans le RE dans le secteur de l'industrie. Pour décrire le processus de RE, quatre grandes étapes sont proposées :

1. l'acquisition / le scan des données ;
2. le pré-traitement / le traitement des points (nettoyage du nuage pour enlever le bruit) ;
3. le maillage et l'extraction des caractéristiques ;
4. la segmentation, la reconnaissance de surface et la modélisation du solide.

Cependant la notion d'assemblage n'est pas abordée dans cette étude.

Conclusion sur la veille technologique et scientifique concernant le RE

Les applications logicielles actuellement sur le marché permettent de reconstruire le modèle CAO à partir de NURBS qui sont ensuite utilisées pour retrouver les esquisses des entités géométriques du composant. Dans ce cas, le modèle 3D final est éditable dans un modeleur CAO alors que dans *3DReshaper*, *Pro Engineer Rev. Eng* ou *Polyworks*, le modèle final est un solide mort. De plus, les assemblages sont très peu traités, seules des pièces uniques le sont. *Geomagic* (et *RapidForm*) et *Catia* proposent une reconstruction avec la création d'un arbre de construction mais chaque entité de ce dernier est indépendante des autres. Dans le cas de modification d'une des fonctions (extrusion par exemple), aucune propagation des modifications aux autres fonctions qui pourraient en hériter, n'est possible. Il faut alors re-modifier toutes celles qui pourraient être impactées.

Concernant les travaux scientifiques traitant du RE, ils abordent chacun un ou plusieurs aspects de nos travaux : [Bénière, 2012] et [Herlem *et al.*, 2012] pour l'extraction d'informations et la signature par graphe ou encore [Wang *et al.*, 2012] pour la reconstruction de modèles CAO paramétrés.

A part les travaux de [Herlem *et al.*, 2012], les études traitent seulement des pièces uniques. L'application du RE à des assemblages est très peu abordée dans la littérature. Néanmoins, les travaux de [Herlem *et al.*, 2012] restent à un niveau de détails faible et ne permettraient pas de reconstruire la maquette numérique avec des modèles CAO paramétrés.

Dans le cadre de nos travaux, aucun logiciel ne peut être utilisé pour répondre à notre besoin. Même si *RapidForm* ou *Catia* présentent des fonctionnalités qui pourraient nous intéresser, aucun de ces deux logiciels ne permet d'identifier (dans le sens de la reconnaissance) un composant qui est "reversé". Enfin concernant les études scientifiques dans le domaine du RE, nous pourrons retenir les travaux de [Deveau, 2006] concernant l'utilisation conjointe de données 2D et 3D et nous verrons si la méthode proposée par [Herlem *et al.*, 2012] peut s'appliquer à notre problématique (pour la comparaison de nomenclature par exemple).

1.5 Synthèse du positionnement de la recherche et énoncé de la problématique

Grâce à la section 1.1, nous avons mis en corrélation notre approche par rapport aux étapes du processus général de RE et la nature des données qui interagissent au fur et à mesure. La partie 1.2 a permis d'identifier trois scénarios de RE dans l'industrie. Enfin la section 1.3 a donné un inventaire des solutions logicielles présentes sur le marché en présentant leur fonctionnement et leurs limites. Puis nous avons présenté les travaux scientifiques traitant du RE et de leur positionnement par rapport à notre approche dans la section 1.4.

Grâce à l'étude de l'existant dans le domaine du RE, on peut alors se poser les questions suivantes :
- puisque le RE peut être utilisé à différentes étapes du cycle de vie du produit, comment pourrait-on intégrer d'autres données hétérogènes inhérentes à la maquette numérique tels que les dessins de définition ou les photographies ? Ces dernières pourraient accélérer le processus qui est d'autant plus fastidieux que l'assemblage est complexe. Elles pourraient également enrichir la progression de la démarche de RE en vue de récupérer des modèles CAO paramétrés.
- comment récupérer les modèles CAO de la maquette numérique d'un assemblage à partir de données incomplètes ? En d'autres termes, comment faire du RE sans avoir à démonter toutes les pièces de l'assemblage et en récupérant de manière automatique des modèles CAO paramétrés ?
- dans un contexte routinier, comment pourrait-on améliorer le processus de RE en minimisant au maximum l'intervention d'un utilisateur ? En effet, on essaye de construire un modèle à partir de la donnée d'entrée (nuage de points) sans chercher à trouver des similitudes avec un autre composant ayant les mêmes caractéristiques géométriques ayant été scanné auparavant et chaque reconstruction de modèle est "unique".

De cela découle la problématique suivante :

> **Quelle méthodologie implémenter afin de récupérer les informations nécessaires à la reconstruction d'une maquette numérique paramétrée d'un assemblage mécanique :**
> - **en utilisant des données hétérogènes ;**
> - **en intégrant les informations d'expertise liées à un métier ;**
> - **le tout dans un contexte routinier ?**

Trois pistes de réflexion seront mises en lumière dans cette thèse :
- la **segmentation des données** : selon le type de données (2D ou 3D) et leur complétude (surfaces non numérisées ou non-visibles sur la vue 2D). En fonction du type de données, quelle technique de segmentation serait la plus efficace afin de reconnaître la topologie et/la géométrie dans la donnée ? il s'agit de trouver un moyen d'extraire le maximum de caractéristiques qui serviront pour l'étape de signature.
- la **signature des données** qui correspond à un ensemble de caractéristiques. C'est un moyen d'identifier les composants mécaniques (nuage de points et photographie) en comparant les similarités entre leur signature et l'ensemble des signatures d'une base de données. Il faudra alors trouver

la signature idéale par rapport au niveau de détails souhaité en sortie (maquette numérique ou simple nomenclature de produit), le type de la donnée (2D ou 3D) ainsi que sa complétude.
— la **comparaison de la signature** avec les autres signatures de la base de connaissances et ceci malgré l'incomplétude des données lors de leur acquisition. Sans démontage, il est impossible de récupérer la géométrie complète du composant que ce soit par photographie ou par la technologie de scan laser. L'**existence d'une base de connaissances** couplée à notre méthodologie accélèrerait le processus en fournissant les informations déjà "reconnues" pour une même pièce. En d'autres termes, si un assemblage est scanné à différentes reprises (à intervalle de temps de plusieurs semaines par exemple), chaque reconstruction ne devra pas se faire "à partir de zéro" à chaque fois. De plus, un composant pourrait être identifié, même partiellement.

1.6 Expérimentation de la proposition et méthodologie de recherche

Afin d'illustrer notre proposition, un cas d'étude simple a été créé comportant l'assemblage d'un piston avec une bielle et d'un vilebrequin. Ce lot de données regroupe les photos de chaque composant (assemblés et séparément) ainsi que les nuages de points correspondants. Ces données ont été fournies par l'Institut Français du Pétrole et des Énergies Nouvelles (IFPEN) qui est également un partenaire industriel. Un consortium de recherche a été créé suite au lancement du projet METIS. Le produit du projet est un démonstrateur logiciel qui porte le même nom (METIS). Pour la méthodologie de recherche, nous nous sommes inspirés des travaux de [Blessing et Chakrabarti, 2009] qui se décompose en quatre étapes :
— la clarification de la recherche (RC) permettant de décrire la situation initiale ainsi que l'idéal souhaité pour la solution finale ;
— l'étude descriptive I (DS-I) qui, grâce à une étude approfondie de la littérature, aide à identifier les facteurs prépondérants sur lesquels il faut agir afin d'atteindre notre idéal ;
— l'étude prescriptive (PS) consistant à développer la solution qui va répondre à l'idéal : différentes techniques de segmentation et de signature ont été testées grâce aux algorithmes développés lors ces trois années de thèse ;
— l'étude descriptive II (DS-II) consiste à évaluer l'impact de la solution développée en mesurant sa capacité à atteindre la situation souhaitée (niveau de performance et limites).

Postulats, hypothèses de recherche et synthèse du positionnement scientifique

Afin d'exposer le contenu de ces travaux, plusieurs hypothèses et postulats doivent être considérés.

Postulat 1 L'activité de RE dont il est question dans ces travaux est routinière, justifiant ainsi la nécessité d'une automatisation de certaines tâches. Ce postulat justifie également la nécessité d'avoir une base de connaissances permettant également de capitaliser au fur et à mesure les connaissances acquises lors du processus de RE. Cette base de connaissances sera propre à un domaine ou orientée selon un référentiel métier propre à l'entreprise dans laquelle se déroule l'activité de RE. Par exemple, il ne serait pas envisageable de faire le RE d'un moteur de voiture si la base de connaissances ne disposait pas des signatures et des modèles CAO des composants du moteur en question. L'indexation [5] de la base de connaissances ne fera pas l'objet de ces travaux. Seule la nature des données qui la composent nous intéresse. On pourra cependant faire l'analogie avec la classification scientifique des espèces utilisée en biologie en imaginant cette base avec des composants mécaniques. Ainsi, comme illustré dans la figure 1.15, nous trouverons dans un moteur (le produit final) plusieurs sous-assemblages comme le groupe motopropulseur qui est lui-même composé de composants tels que les bielles et les pistons. En contexte d'utilisation, notre méthodologie devra traiter des assemblages de plusieurs milliers de composants comme dans un moteur d'automobile par exemple.

5. Indexation : elle est définie par [Tangelder et Veltkamp, 2007] comme le processus permettant de construire une structure des données afin d'accélérer la recherche (dans la base de connaissances).

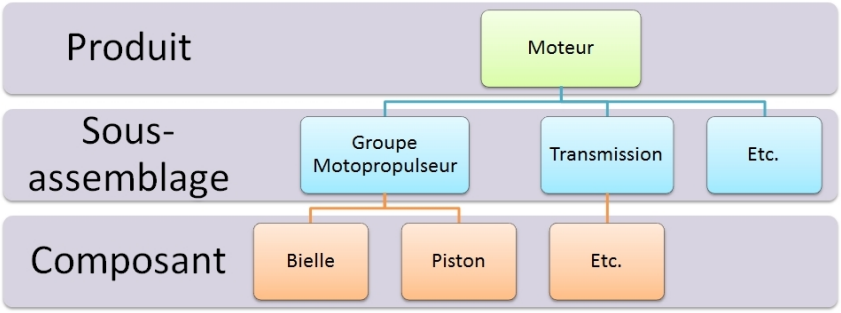


Figure 1.15 – Classification des composants dans un moteur automobile.

Afin de constituer un ensemble de données suffisamment important et du fait de l'impossibilité de scanner directement autant d'assemblages divers que variés, il a été décidé d'utiliser les modèles CAO d'assemblages et de composants seuls qui ont ensuite été maillés et enregistrés en .stl (grâce au logiciel CATIA^{TM}, atelier *STL Prototypage Rapide*).

La base de connaissances contient également des modèles CAO paramétrés de ce moteur appelés *template*. Leur paramétrage et l'ensemble des connaissances qu'ils contiennent ne fait pas partie de cette étude. Il ne s'agira pas de remettre en question la pertinence de ces informations. Cependant, elles nous guideront afin de savoir quel type d'information sont à extraire des données d'entrée et comment interpréter les signatures en vue de reconstruire le modèle CAO.

Postulat 2 Le deuxième postulat concerne la notion de données hétérogènes. En effet comme expliqué précédemment, le processus de RE présenté dans ces travaux se différencie par son aptitude à utiliser des données de différentes dimensions :

— 3D pour les nuages de points, les maquettes numériques et leurs composants ;
— 2D pour les dessins de définition et les photographies ;
— 0D pour les documents socio-techniques tels que les manuels de maintenance.

La figure 1.16 relate les principaux types de données possibles en ce qui concerne le domaine de l'ingénierie mécanique. Cette liste a été proposée dans le cadre du projet METIS, sur lequel reposent nos travaux.

Pour ce qui concerne nos travaux, nous ferons appel uniquement aux nuages de points (STL issus de scan sans contact), aux photographies et aux dessins techniques (dessin de cotation fonctionnelle) ; les nuages de points gardant une place importante tout au long de cette étude. Les données qui seront utilisées dans notre processus de RE seront déjà numérisées pour ce qui concerne les photographies et les nuages de points.

Pour les photographies, le format standard .jpeg ou .bmp est considéré. L'acquisition de ces photos ne fait pas l'objet de ces recherches.

Pour les nuages de points, nous considérons l'utilisation d'un bras de mesure laser portable tels que le Handyscan^{TM} ou METRIS^{TM}. L'obtention du nuage de points sous le format .stl (binaire ou ASCII) est effectué grâce au logiciel propre au matériel disponible. Nous utilisons le nuage "brut" sans aucune modification (pas de remplissage des trous). Seul un nettoyage grossier du nuage pour enlever des points inutiles est envisagé, en enlevant par exemple les points concernant le support sur lequel est posé l'assemblage (table ou autre support). Un des buts de ces travaux est d'aider l'utilisateur dans sa démarche de RE, en limitant notamment les opérations de pré-traitement des données, en utilisant au maximum, les données à l'état brut.

Ensuite nous considérerons aussi l'existence de maquette numérique ou nomenclature des produits comme données d'entrée qui, selon le cas industriel ("*As designed / as made*" ou "*As designed / as maintained*"), fourniront des informations afin de reconstruire le modèle 3D final.

Hypothèse 1 Concernant l'étape de signature des données, nous ferons l'hypothèse qu'il existe au moins autant de mécanismes de signature que de types de données (nuages de points, images, maquette numérique etc.).

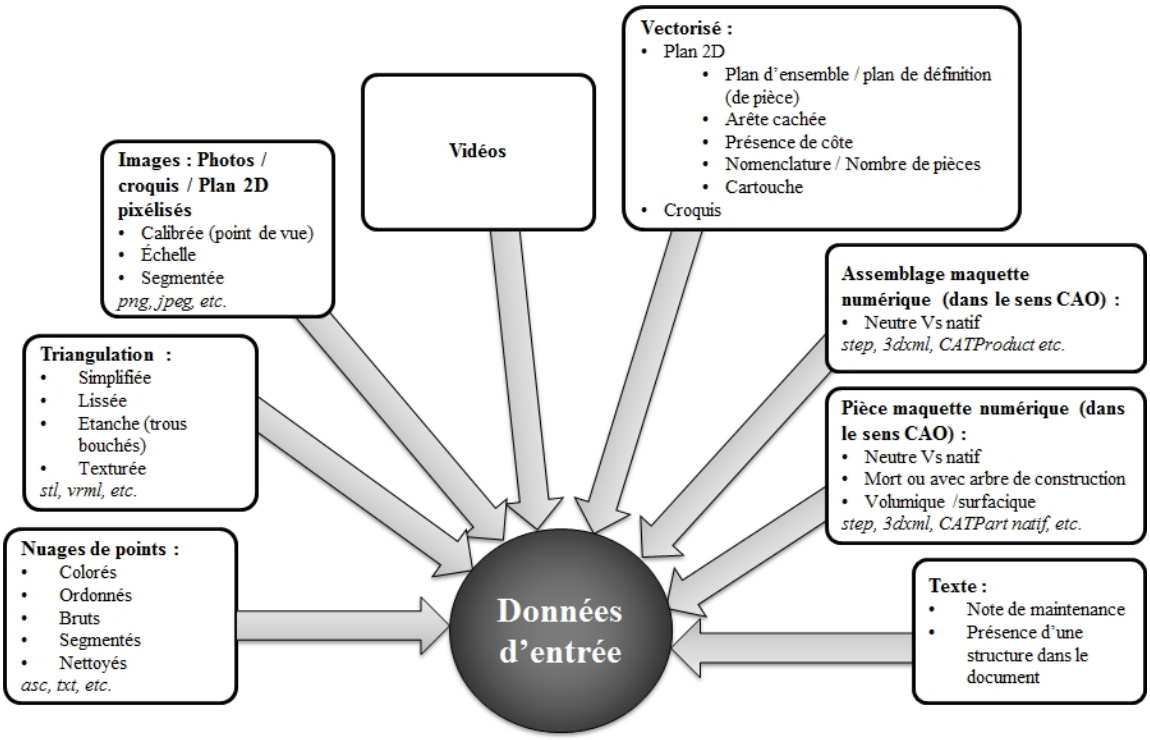


Figure 1.16 – Liste non-exhaustive des différents types de données utilisés en ingénierie mécanique de nos jours.

Postulat 3 En sortie de notre processus de RE, nous envisagerons trois niveaux de détails pour la reconstruction du modèle numérique :

— un niveau global : avec une nomenclature simple de l'assemblage mécanique (niveau de détails le plus bas) ;
— un niveau topologique et géométrique : une nomenclature avec les enveloppes externes des solides ;
— un niveau fonctionnel : une maquette numérique de l'assemblage avec des modèles CAO paramétrés (niveau le plus haut).

La figure 1.17 fait la synthèse du contexte général de nos travaux. Nous retrouvons en entrée trois types données : les maquettes numériques (CAO), les nuages de points (format .stl) ainsi que les images (formats .bmp ou .jpeg). La méthodologie s'appuiera sur le principe de signature globale, géométrique et topologique ou fonctionnelle en fonction de chaque type de donnée. L'ensemble des données d'entrée sont d'abord segmentées, puis signées. Leurs signatures relatives sont ensuite comparées avec celles présentes dans la BDC. Les résultats de cette comparaison permettent d'identifier les composants mécaniques présents dans les données d'entrée. Pour réaliser cela, la base de connaissances (BDC) comporte :

— un ensemble de signatures de différents types (2D ou 3D) ;
— des modèles CAO paramétrés appartenant au domaine d'application du RE permettant la reconstruction de la maquette numérique.

La figure 1.17 servira de fil rouge tout au long de ces travaux.

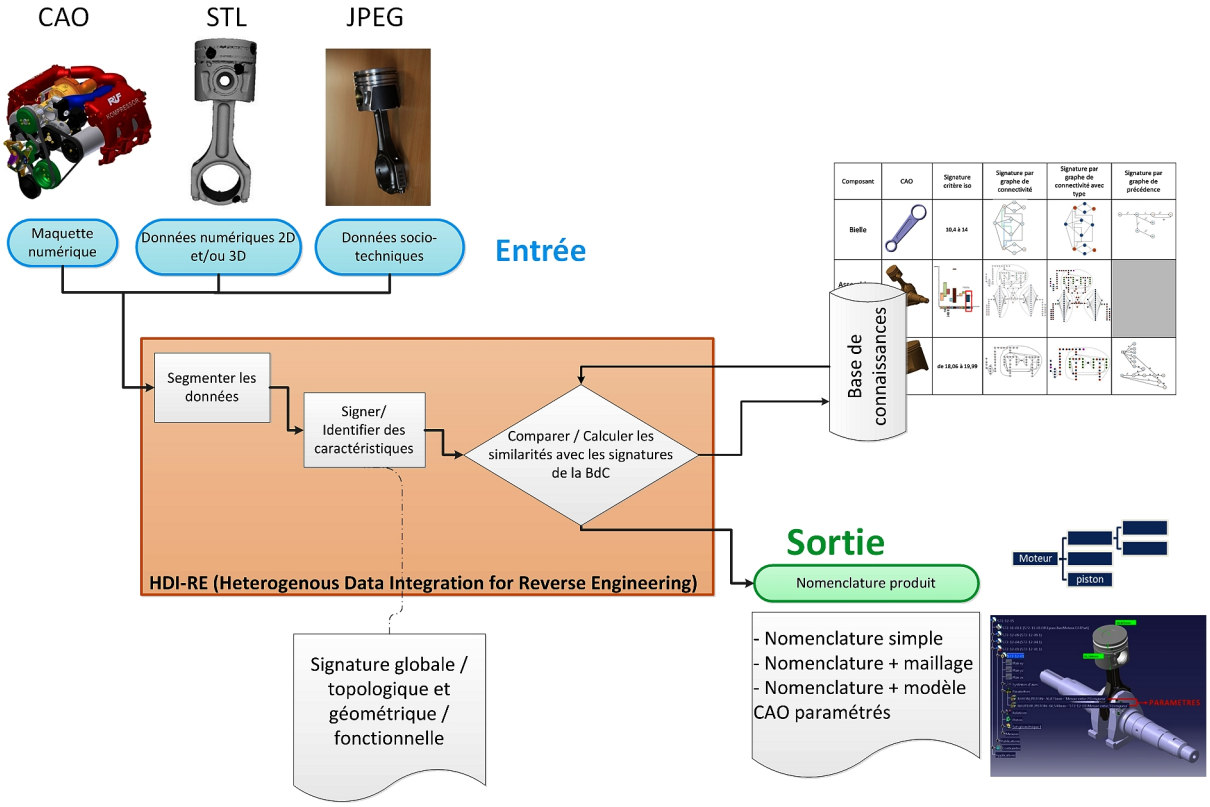


Figure 1.17 – Hypothèses de départ et proposition de la méthodologie HDI-RE.

Conclusion

Ce chapitre nous a permis de mettre en évidence les différents cas industriels dans lesquels le Reverse Engineering joue un rôle important dans des tâches routinières de contrôle de la qualité, de conformité ou encore dans les phases amont du développement de produits. Les logiciels existants répondent difficilement au contexte de RE routinier ainsi qu'au problème d'incomplétude des données. Notre proposition prend en compte ces manques par rapport à l'offre actuelle du marché.

Nos travaux se positionnent ainsi sur les trois étapes du processus de RE :

1. la segmentation des données ;
2. la signature des données ;
3. la comparaison de signatures grâce à la présence d'une base de connaissances ;

le tout en intégrant les notions de données hétérogènes et de modèles CAO paramétrés.

Le chapitre suivant apportera un état de l'art sur chaque étape citée ci-dessus.

CHAPITRE 2

État de l'art : le Reverse Engineering, entre Shape Matching et Knowledge-Based Engineering

"N'essayez pas de devenir un homme qui a du succès. Essayez de devenir un homme qui a de la valeur."

- Albert Einstein

Introduction

Le premier chapitre a permis d'identifier trois pistes de recherche afin de répondre à la problématique de cette étude. Il s'agira de rechercher une ou plusieurs solutions permettant d'intégrer des données hétérogènes dans un contexte de RE routinier dans le but de récupérer en sortie une maquette numérique contenant des modèles CAO paramétrés. Ces solutions doivent aider l'utilisateur à identifier automatiquement les différentes pièces présentes dans les données d'entrée et extraire toutes les informations géométriques et/ou topologiques nécessaires en vue de reconstruire la maquette numérique.

Ce chapitre propose l'état de l'art suivant :

— section 2.1 nous aborderons les travaux dans le domaine de segmentation et cela pour les données 2D puis 3D ;

— section 2.2 nous étudierons ensuite les recherches scientifiques concernant les signatures pour les données 2D et 3D. Elles constituent le moyen de formaliser les informations extraites des données, c'est-à-dire les connaissances ;

— section 2.3 nous regarderons les résultats des études dans le domaine de la gestion des connaissances (KBE) afin de capitaliser et réutiliser les connaissances acquises au fur et à mesure de l'activité de RE. Nous verrons ensuite les travaux traitant de la comparaison des signatures.

2.1 La segmentation des données

Il s'agit de la première étape de notre processus de RE.

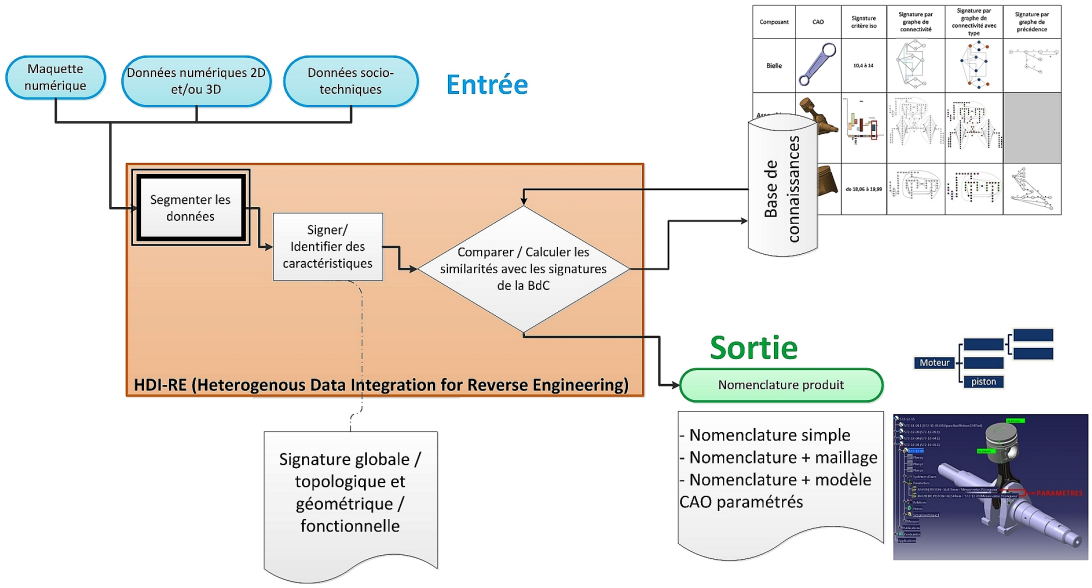


Figure 2.1 – Première étape de HDI-RE : la segmentation.

Suivant le domaine et le type de données, il existe différentes définitions à la segmentation ainsi que différentes techniques de segmentation. Le domaine est très large que ce soit pour les données 3D ou les données 2D. La segmentation des données n'est qu'une étape de notre processus de RE. Ainsi nous aborderons les études les plus pertinentes en les classant par type ou technique.

2.1.1 La segmentation des images (données 2D)

La segmentation d'image est très liée à la reconnaissance de forme. En effet, en fonction du critère qui sert à comparer puis reconnaître une forme dans une image, un type de segmentation correspond. Par exemple, si nous souhaitons reconnaitre une forme (un cercle) alors que nous utilisons une base de connaissances possédant une multitude de contours (rectangle, ligne, cercle, ellipse etc.) alors, nous utiliserons une technique de segmentation permettant d'extraire les contours des formes présentes dans cette image. Bien des travaux traitent directement de la reconnaissance de forme et y incluent la segmentation des images. Nous tâcherons de les distinguer même si la limite entre ces deux domaines est faible.

De nombreuses études ont permis d'identifier toutes les techniques de segmentation. [Spirkovska, 1993] propose de les classer en trois catégories :

1. la segmentation basée sur les pixels ;
2. la segmentation basée sur les arêtes ;
3. la segmentation basée sur les régions.

[Kermad, 1997] les nomme comme des approches "non coopératives" et propose de les distinguer des approches "coopératives" qui sont en fait une combinaison des différentes techniques citées dans la liste ci-dessus.

Dans la littérature, il existe plusieurs manières de classer les techniques de segmentation : par classe (pixel, arête, région) ou par type de segmentation car certaines techniques sont applicables à différentes classes. Grâce aux états de l'art sur la segmentation de [Deveau, 2006; Kermad, 1997] ainsi qu'à l'étude de [Spirkovska, 1993], nous proposons une classification par pixels/arête/régions que nous présentons dans le tableau 2.1 qui synthétise les informations de ces trois études.

Approches non coopératives			Approches coopératives et autres
Pixels	Arêtes	Régions	Pixels / Arêtes / Régions
Seuillage [Kermad, 1997, section 2.2.1.2] : • locale • globale	Détection de contours ou de frontières : • opérateur 1er ordre : gradient, filtre de Sobel [Deveau, 2006, section 2.1.1] • opérateur 2nd ordre : Laplacien [Deveau, 2006, section 2.1.1]	Approche statistique [Kermad, 1997, section 2.2.3.1] : • extraction de paramètres et classification • modélisation	Approches séquentielles, parallèles et hybrides [Kermad, 1997, section 2.3] RANSAC (RANdom SAmple Consensus) [Deveau, 2006, section 2.1.4] Segmentation et fusion de données [Deveau, 2006, section 2.1.5]
Classification [Deveau, 2006, section 2.1.3]	• modélisation surfacique [Kermad, 1997, section 2.2.2.3] • par optimisation : filtre de Canny, Deriche, Shen et Castan [Kermad, 1997, section 2.2.2.4] • varationnelle [Kermad, 1997, section 2.2.2.5] ou modèles actifs ("snakes") [Deveau, 2006, section 2.1.1] • statistique [Kermad, 1997, section 2.2.2.5] • programmations dynamiques, morphologie mathématique etc. [Kermad, 1997, section 2.2.2.5] Fermeture de contours : • recherche du meilleur chemin à partir d'une extrémité [Kermad, 1997, section 2.2.2.6] • groupement perceptuel [Kermad, 1997, section 2.2.2.6]	Approche structurelle [Kermad, 1997, section 2.2.3.1] Statistique bayésienne [Deveau, 2006, section 2.1.2] Croissance de régions (aussi appellé clustering ou aggrégation de points) [Deveau, 2006, section 2.1.2] : • critère de distances • critères de proximité entre courbures ou normales estimées localement Découpe de région / fusion • slip/merge (division / fusion de régions) [Kermad, 1997, section 2.2.1.3.4] [Deveau, 2006, section 2.1.2]	Segmentation hiérarchique [Deveau, 2006, section 2.2]

Tableau 2.1 – Classification des techniques de segmentation pour les données 2D.

2.1.1.1 La segmentation basée sur les pixels

Dans les techniques de segmentation utilisant les pixels, nous retrouvons les méthodes appelées de **seuillage**. L'image est considérée comme une donnée bidimensionnelle : chaque pixel de l'image correspond à un niveau d'intensité lumineuse (niveau de gris). [Kermad, 1997] définit alors l'histogramme h des niveaux de gris comme étant la "fonction qui associe à chaque niveau de gris g_i , le nombre de pixels de l'image $h(g_i)$ qui possèdent cette intensité lumineuse".

FIGURE 2.2 – Histogramme bi-modal d'une image [Spirkovska, 1993] : le mode (ap) représente les pixels de l'arrière-plan et l'autre représente les pixels de l'objet.

Il existe plusieurs algorithmes qui permettent de faire du seuillage d'histogramme, c'est-à-dire, ce qui permet de déterminer le seuil afin de définir le niveau de gris entre deux modes comme nous pouvons l'observer dans la figure 2.2. [Spirkovska, 1993] teste deux algorithmes : celui d'Otsu et celui par un filtre Gaussien. Les travaux de [Sri Madhava Raja *et al.*, 2014] proposent une comparaison de plusieurs critères appliqués à l'algorithme d'Otsu et ils sont testés sur des images appartenant au jeu de données Berkeley ([Martin *et al.*, 2001]).
Enfin [Kermad, 1997, section 2.2.1.2] propose de séparer les méthodes de seuillage en deux catégories : les méthodes locales et les globales. La figure 2.3 relate les différentes méthodes présentées dans ses travaux. Dans les méthodes globales, une seule valeur de niveau de gris est fixée pour l'ensemble de l'image. Quant aux méthodes locales, elles consistent à "déterminer en chaque point un seuil dépendant de l'histogramme de répartition des luminances [1] de son voisinage".

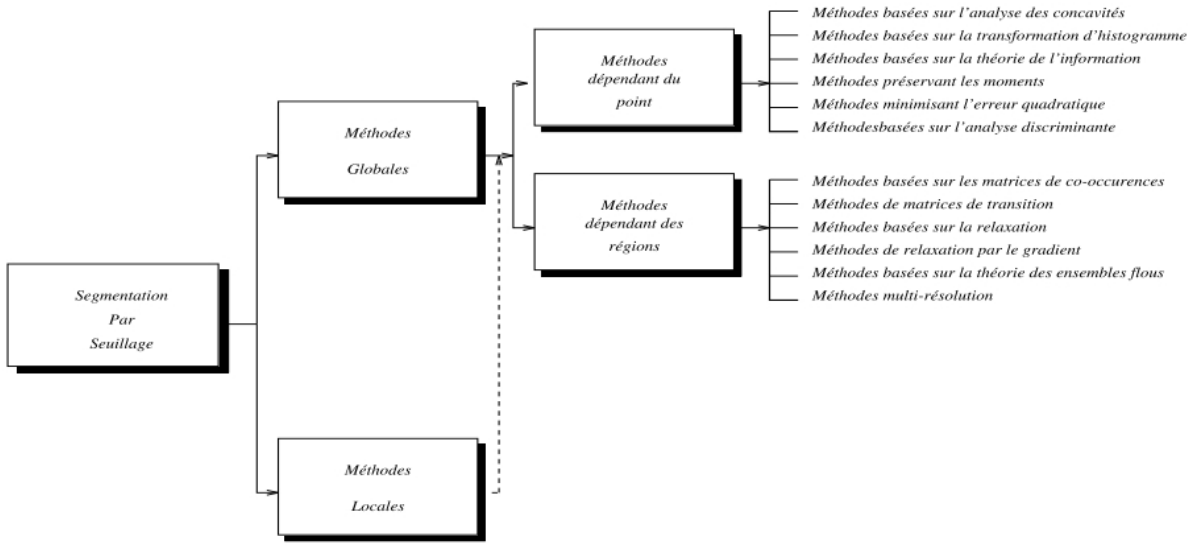


FIGURE 2.3 – Méthodes de segmentation par seuillage proposées par [Kermad, 1997, section 2.2.1.2]

Comme technique de segmentation, nous retrouvons aussi la **classification** qui selon [Deveau, 2006] permet de regrouper dans une même classe et selon un critère déterminé, des éléments présentant des caractéristiques comparables. La transformation de [Hough, 1962] fait partie de ces méthodes. Nous remarquerons que les techniques de classification sont aussi utilisées dans les méthodes basées sur les régions.

1. Luminance : c'est une grandeur mesurable dans le domaine de la photométrie correspondant à la sensation visuelle de luminosité d'une surface.

2.1.1.2 La segmentation basée sur les arêtes

D'après [Spirkovska, 1993], les techniques basées sur les arêtes, aussi appelées **approches frontières**, consistent à "détecter les contours dans une image et à utiliser cette information afin de séparer les régions". [Kermad, 1997] ajoute que ces méthodes aident à détecter les discontinuités des intensités lumineuses, c'est-à-dire lorsqu'il y a une forte transition des valeurs d'intensité.

Pour les arêtes, une autre technique est celle appelée par **fermeture de contours** [Kermad, 1997, section 2.2.2.6]. Elle est utilisée par exemple pour l'exploration de graphes. Dans cette catégorie, nous retrouvons la méthode dite "snake"(serpent) où une courbe (polynomiale ou paramétrée) évolue à partir d'une position initiale et cela à proximité des frontières à détecter. Cette méthode est décrite dans les travaux de [Deveau, 2006, section 2.1.1].

Parmi les travaux les plus connus dans les techniques de segmentation basées sur les arêtes, nous citerons les filtres de [Canny, 1986; Sobel, 1978] qui servent de référence dans la littérature. L'étude de [Othman *et al.*, 2009] permet de les comparer. A partir d'une image IRM d'un genou, ils souhaitent réduire la quantité de données en filtrant l'image et ceci afin de récupérer les contours des os et des tissus. Comme nous pouvons l'observer entre les images (d) et (e) de la figure 2.4, le filtre de Canny donne un meilleur résultat pour l'extraction des contours. En effet, le filtre de Sobel ne peut fournir des arêtes lisses et fines, de plus elles sont discontinues. Selon [Othman *et al.*, 2009] cette différence serait dû à la présence de bruit dans l'image. Le filtre de Canny semble le plus performant cependant le calcul nécessiterait une mémoire d'ordinateur importante.

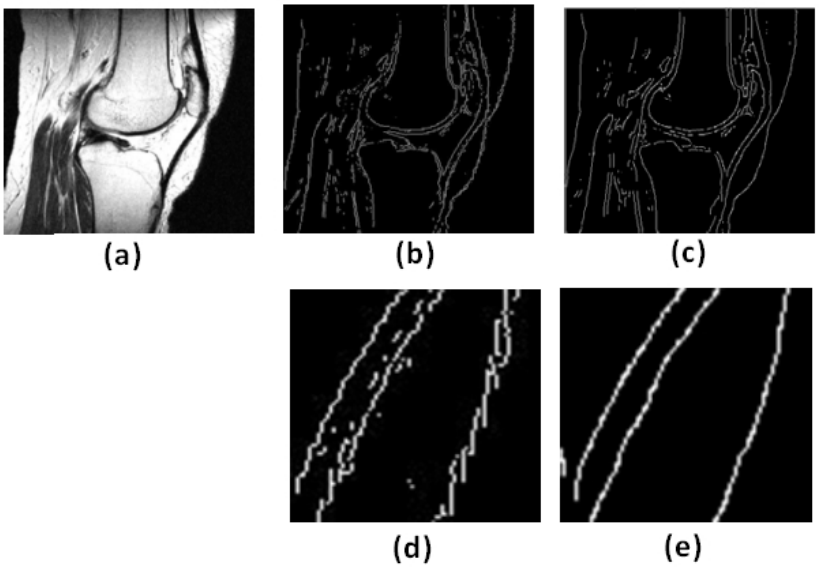


Figure 2.4 – Étude comparative des filtres de Sobel et Canny [Othman *et al.*, 2009] : (a) image IRM, (b) détection des contours avec le filtre de Sobel, (c) résultats de la détection avec le filtre de Canny, (d) contours générés par Sobel et (e) contours obtenus avec Canny.

2.1.1.3 La segmentation basée sur les régions

Nous avons d'un côté [Spirkovska, 1993] qui définit les régions comme des zones avec des pixels ayant les mêmes caractéristiques (même niveau de gris par exemple). Elle présente deux types de méthodes : celles par croissance de régions, celles par découpe de régions. La deuxième se caractérise par le fait de commencer par l'image entière tel un grain. Si le grain n'est pas homogène, alors il est découpé en un nombre pré-déterminé de sous-régions, généralement quatre. Puis on re-teste l'homogénéité de chaque grain jusqu'à ce que chaque sous-région soit homogène. D'après [Spirkovska, 1993], la technique par découpe de régions serait moins sensible au bruit que celle par croissance de régions.

D'un autre côté, [Kermad, 1997] répertorie deux types de régions : les régions non texturées et les régions texturées. Il définit les **régions non texturées** comme des ensembles de points qui partagent des propriétés similaires : pixels de niveaux semblables sans qu'ils soient connexes. Ces régions sont segmentées grâce à deux méthodes : par classification (seuillage d'histogramme) ou par croissance de régions [2].

Quant aux **régions texturées**, il les définit comme des zones non homogènes au sens des niveaux de gris. La texture serait une "région de l'image dont l'observation se traduit par une impression visuelle d'homogénéité pour toutes les transitions possibles à l'intérieur de cette région et se traduisant par une répartition spatiale d'un même motif dans différentes directions de l'espace". Ce serait le cas des images satellitaires, des images médicales pour citer que quelques exemples. Il présente également une classification de toutes les approches de segmentations des régions texturées (figure 2.5).

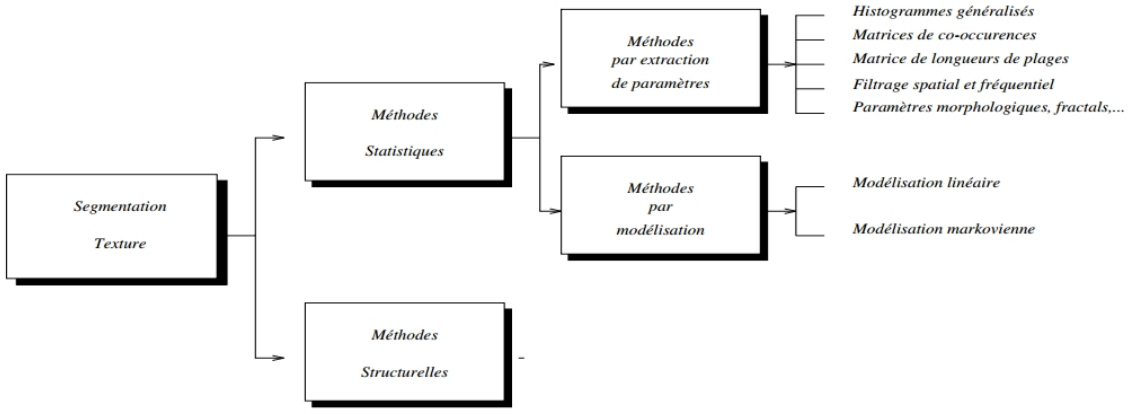


Figure 2.5 – Classification des méthodes de segmentation de régions texturées proposée par [Kermad, 1997].

2.1.1.4 Les autres méthodes dites coopératives

Les méthodes coopératives sont, comme cité précédemment, des approches mélangeant deux ou trois des méthodes présentées dans les sections 2.1.1.1, 2.1.1.2 et 2.1.1.3. [Kermad, 1997] en répertorie trois catégories :

- les approches séquentielles ;
- les approches parallèles ;
- les approches hybrides.

On retrouve la définition et les travaux relatifs de chacune d'entre elle dans [Kermad, 1997, section 2.3].

Dans les autres méthodes (exceptées les trois citées ci-dessus), nous pouvons citer RANSAC (RANdom SAmple Consensus) de [Fischler et Bolles, 1981] qui est également utilisée pour la segmentation des données 3D. Cette méthode est un estimateur de formes et en même temps un outil de segmentation. L'approche est itérative et consiste à choisir de manière aléatoire un échantillon de pixels. Après estimation de nombre d'éléments composants l'échantillon, on vient appliquer un modèle (un plan par exemple) et on calcule la distance des points de l'échantillon par rapport au modèle. Si un pourcentage suffisant de points de l'échantillon appartient au modèle, alors on augmente la taille du modèle et cela de manière itérative. Le modèle dont le support est le plus grand est gardé. Puis on passe à un autre échantillon.

Nous pouvons citer également l'étude de [Deveau, 2006] où il présente la segmentation hiérarchique et dont l'objectif est de construire une hiérarchie de régions basée sur le calcul d'une énergie globale qui doit être la plus petite possible. Cette dernière fait intervenir deux termes : un terme de régularisation

2. Croissance de régions : [Kermad, 1997] définit cette méthode par la "sélection d'un pixel ou d'un ensemble de pixels de l'image, appelé noyau, autour duquel on fait croître une région en sélectionnant les pixels qui satisfont un critère de similitude".

C et un terme d'attache aux données D, pondérés par un paramètre d'échelle λ. L'énergie définie pour une partition P de la scène s'écrit :

$$E_\lambda(P) = \sum_{R \in P} \lambda C(R) + D(R) \tag{2.1}$$

La hiérarchie des régions se construit par l'augmentation du paramètre d'échelle λ, en fusionnant les régions qui font décroître l'énergie globale $E_\lambda(P)$. Un graphe d'adjacence sert ensuite de structure ; chaque nœud représentant une région et chaque arête reliant les régions adjacentes. Le graphe qui en découle est appelé graphe hiérarchique. L'approche est illustrée dans la figure 2.6. "Le graphe permet de

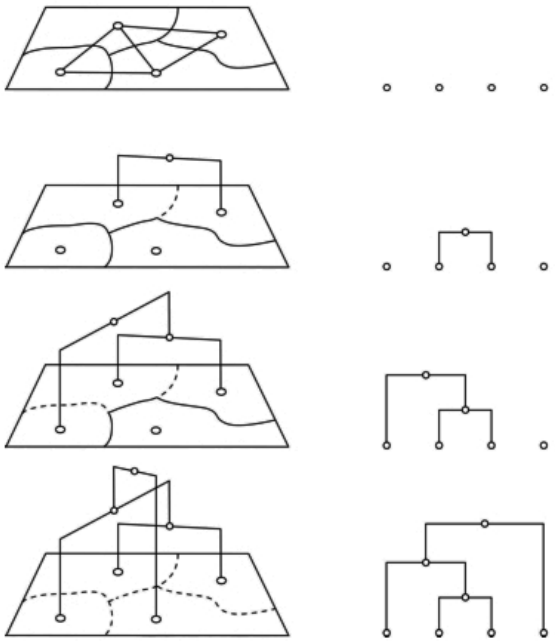


Figure 2.6 – Principe de construction du graphe hiérarchique par [Deveau, 2006].

gérer la fusion des régions : les nœuds du graphe sont affectés d'un attribut qui correspond à l'énergie de la région, une arête du graphe contient une valeur associée à l'énergie de fusion de deux régions adjacentes. La construction du graphe hiérarchique se fait par fusion itérative des régions en fonction de leur énergie. Il s'agit de considérer l'ensemble des énergies affectées aux arêtes, de choisir l'arête dont l'énergie est la plus faible, puis de fusionner les régions reliées par cette arête". Le calcul de ces énergies est décrit précisément dans [Deveau, 2006, section 2.3.1].

Dans la même catégorie, nous retrouvons les travaux de [Chang et Park, 2001] qui utilisent deux types d'images. Les images d'intensité (photographie) et les images de profondeur, qui sont obtenues par un laser et un capteur qui calculent la distance de l'objet par rapport au capteur. L'image qui en résulte possède également des pixels dont la valeur correspond à une distance. Ces travaux relatent d'une méthodologie de fusion entre ces deux types d'image et qui permet d'extraire les contours des objets malgré le bruit.

Conclusion sur la segmentation des images

L'étude des différentes techniques de segmentation des images a permis de mettre en évidence le type de caractéristiques (segments) dont nous aurions besoin dans le cadre de nos travaux. De toute évidence, l'exploitation directe des pixels ou des arêtes ne fournit pas un niveau d'information suffisant qui puisse nous servir dans la démarche de RE. En effet, les contours d'un objet dans une image se reportent à une vue. Afin de pouvoir identifier (étape de comparaison des signatures), il faudrait par la suite posséder une base de données de toutes les vues possibles de chaque objet ainsi que de chaque assemblage ce qui ralentirait le temps de processus.

Cependant pour les approches basées sur les régions, les régions segmentées peuvent aider à délimiter des régions dans les nuages de points issus de scan qui, comme nous le verrons dans la section 2.1.2, seront également segmentées par région. C'est ce que montre [Deveau, 2006] dans ses travaux. Les images permettent de segmenter des nuages de points en délimitant les limites entre composants.

Enfin parmi les approches coopératives, la méthode RANSAC proposée par [Fischler et Bolles, 1981] s'avère très utile car l'étape de segmentation permet également de définir le type de forme (cercle, droite). Cette information serait utile pour l'étape de signature.

2.1.2 La segmentation des données 3D

Les techniques de segmentation pour les données 3D sont assez similaires que pour les données 2D telles que les images. Cette section sera traitée de la même manière que la précédente en évitant toute redondance. Nous considérons les maillages issus de nuage de points (scans) et les modèles CAO comme des données 3D pouvant être segmentées tel que défini dans le chapitre 1.

[Woo *et al.*, 2002] proposent trois catégories pour classifier les méthodes de segmentation :
— les techniques basées sur les arêtes ;
— les techniques basées sur les régions ;
— les techniques hybrides.

Pour [Attene *et al.*, 2006] , les méthodes de segmentation sont classées en deux catégories :
— celles basées sur la détection de caractéristiques : le but est de déduire des surfaces par extraction de lignes caractéristiques et en découpant la surface en un réseau de caractéristiques semblables ;

— celles basées sur la détection de régions : une estimation avec des approximations des primitives est réalisée. Deux sous-catégories sont ensuite définies. Tout d'abord, il existe les méthodes par croissance de région. Elles concernent notamment les données de type maillage qui comportent des triangles. Cette technique consiste à partir d'un germe à le faire grossir en agrégeant les triangles voisins possédant des caractéristiques similaires au germe. Puis nous retrouvons les méthodes "top-down" où l'on part de l'ensemble du maillage et on vient le fractionner en régions plus petites qui peuvent être mieux approximées par des primitives.

[Shamir, 2008] propose une analyse complète des techniques de segmentation des maillages qu'il propose de classer en deux catégories : les segmentations en fonction du type de surface et les segmentations en fonction du type de portion (morceau de maillage découpé) . Il synthétise tout cela dans un tableau qui regroupe les techniques ainsi que les références des travaux.

Nous choisirons de reprendre la classification de [Woo *et al.*, 2002] car elle est similaire à celle proposée dans la section 2.1.1 et nous y ajouterons une catégorie dédiée aux méthodes basées sur les points pour ce qui concerne les nuages de points. Afin de synthétiser toutes les techniques abordées ci-après, nous proposons le tableau 2.2.

Approches non-coopératives			Approches coopératives et autres
Points	Arêtes	Régions	Points / Arêtes / Régions
Estimation de vecteur normal [Zhan et al., 2010]	Détection d'arêtes : • transformée de Hough [Xiao et al., 2011] • modèle de courbe basée sur des points [Liu et al., 2003] • estimation de la courbure de surface [Fan et al., 1987], [Yang et Lee, 1999] • opérateurs du premier et second ordre [Yang et Lee, 1999] Contours actifs [Milroy et al., 1997]	Croissance de région (aussi appellée "croissance de germe" par [Liu et Ma, 1999]) Inventaire réalisé par [Vo et al., 2015] Croissance de région par source multiple [Lévy et al., 2002] Méthodes implicites [Shamir, 2008] Ligne de partage des eaux par [Delest et al., 2007] Clusterisation : • hiérarchique par [Woo et al., 2002] / [Attene et al., 2006] • itérative par [Liu et Siegwart, 2013] Méthodes d'analyse spectrale par [Delest et al., 2007]	Détection de régions et d'arêtes [Sunil et Pande, 2008] Classification par densité de points puis par courbure de Gauss et transformée de Hough [Xiao et al., 2011] Procédé de multi-résolution par une gamme de tailles de voisinage [Chaperon, 2002]

Tableau 2.2 – Classification des techniques de segmentation pour les données 3D.

2.1.2.1 La segmentation basée sur les points

Peu de travaux existent sur ces techniques car elles se rapprochent très fortement des méthodes basées sur les régions. Et effet, les techniques de croissance de régions peuvent être classifiées avec les techniques basées sur les points puisqu'elles impliquent la sélection d'un germe [3] initial.

La méthode de segmentation des travaux de [Zhan *et al.*, 2010] est basée sur le calcul de vecteurs normaux et sur le groupement de couleurs (clusters de couleurs) appliqué au domaine du LiDAR (*Light Detection And Ranging* qui se traduit par télédétection par laser). Les données (nuages de points) sont obtenues par scan laser aérien.

Leur méthodologie de segmentation est à la fois "géo-métrique" et "colori-métrique". Elle consiste à ajuster un plan sur un ensemble de points du nuage. La sélection de ces points se fait par la méthode des k plus proches voisins (*k-nearest neighbours* - voir définition dans [Collomb, 1980]) avec pour paramètre la distance $d = 0,5m$ et $n = 30$ pour le nombre de voisins à chercher. La méthode des moindres carrés est ensuite appliquée afin de déterminer le plan idéal qui s'ajuste à la surface représentée par l'ensemble de ces points. Le vecteur normal de chaque point par rapport à ce plan est ensuite calculé. Les valeurs des composantes de ce vecteur (= distance au plan) sont associées à une couleur (les coordonnées x, y et z étant utilisées pour les valeurs Rouge, Vert et Bleu de la teinte). On retrouve le résultat de cette première segmentation dans l'image (b) de la figure 2.7. [Zhan *et al.*, 2010] utilisent alors la distance euclidienne pour mesurer la différence entre chaque couleur. Suivant un seuil défini, si deux points sont considérés comme similaires alors, ils sont enregistrés comme appartenant à la même région. Cette deuxième opération permet d'affiner la segmentation notamment par rapport aux points qui auraient pu être considérés comme appartenant au même plan. Nous remarquerons sur l'image (c) de la figure 2.7 que cette segmentation est plus fine : des plans ont été segmentés en plusieurs.

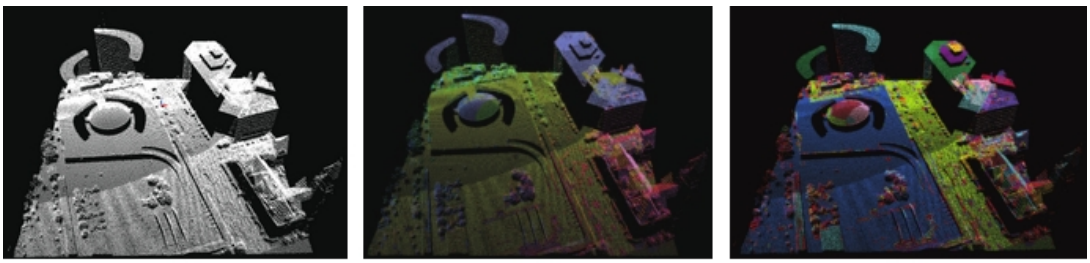


FIGURE 2.7 – Segmentation de nuage de points de bâtiments [Zhan *et al.*, 2010] : (a) le nuage de points brut, (b) le nuage de points avec les vecteurs normaux colorés et (c) le nuage de points après calcul des distances entre les couleurs.

2.1.2.2 La segmentation basée sur les arêtes

Ces techniques sont basées sur la localisation des arêtes, ce qui consiste selon [Woo *et al.*, 2002] à détecter les discontinuités sur les surfaces. Un composant est identifié lorsque un ensemble de contours fermés est détecté dans le nuage de points. L'utilisation des propriétés de courbure locale de surface permet à [Fan *et al.*, 1987] de détecter des contours dans l'ensemble des points. Pour cela, ils partent d'un ensemble de points distincts par lesquels ils font passer des courbes. Ces dernières sont alors classifiées selon leurs propriétés physiques comme par exemple les sauts de frontières, les pliures, les arêtes de crête. Ensuite les pliures et les sauts de frontières sont utilisés pour segmenter les surfaces en différentes pièces.

[Yang et Lee, 1999] proposent de classer les méthodes basées sur les arêtes en trois groupes :
— les opérateurs locaux de premier et second ordre (gradient et Laplacien) ;
— les courbures de surfaces (courbure de Gauss) ;
— les contours de régions qui consistent à découper les données afin de créer des contours d'équi-profondeur (voir travaux de [Wani et Batchelor, 1994]).

3. Germe : ensemble de points 3D ayant les mêmes caractéristiques géométriques ; par analogie avec le niveau de gris des images.

La méthode proposée par [Yang et Lee, 1999] appartient au deuxième groupe ; elle fait appel à une méthode des moindres carrés permettant d'ajuster un plan en un point du nuage. Puis la distance euclidienne des points voisins est calculée, à partir de laquelle il est possible de déduire la courbure de la surface à segmenter. Les propriétés de courbure permettent alors de détecter les "points de bord" (arête entre deux surfaces). La figure 2.8 illustre les différents types d'arête pouvant être détectés sur un nuage de points.

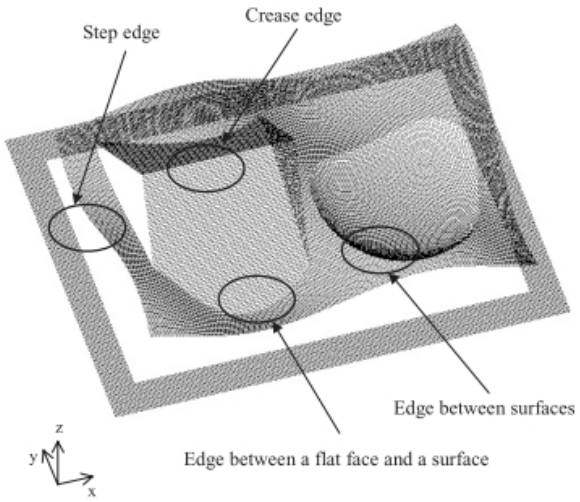


Figure 2.8 – Exemple de détection d'arête de bord par calcul de courbure des surfaces sur un nuage de points [Yang et Lee, 1999].

2.1.2.3 La segmentation basée sur les régions

Ce type de technique est le plus employé dans le cadre du Reverse Engineering. Une région est considérée comme un ensemble d'entités voisines les unes des autres : des triangles dans le cas de maillages, des faces dans le cas de modèles CAO.

[Shamir, 2008] propose de les classer en sept catégories :

1. par croissance de région : cette technique consiste à partir d'un germe (cluster [4] initial) et de le faire croître avec les points voisins. Le germe initial peut être choisi de manière aléatoire ou déterminé à partir d'un ensemble de caractéristiques géométriques. Un critère détermine si tel voisin peut être ajouté au cluster existant. Le choix du germe initial et du critère d'agrégation ont fait l'objet de très nombreux travaux, [Vo *et al.*, 2015, section 2.2] en propose un inventaire. Dans ses travaux, il choisit de grouper les points par adjacence et suivant leurs caractéristiques. Il calcule pour cela la moyenne quadratique [5] des distances orthogonales entre chaque point par rapport à un plan ajusté au mieux sur le maillage.

2. par croissance de région par sources multiples : la différence avec la méthode précédente est le fait que la segmentation débute à partir de plusieurs germes que l'on fait croître en parallèle. Ce type de segmentation apporte un gain de temps considérable. [Shamir, 2008] cite les travaux de [Lévy *et al.*, 2002] qui traitent d'une méthode nommée *Texture Atlas Generation* qui consiste à extraire en premier les contours caractéristiques et à les utiliser comme frontières ; les germes y croissant à l'intérieur.

3. par clusterisation hiérarchique : les techniques basées sur la hiérarchisation sont assez similaires à celles par croissance de région à la différence de leur critère de fusion et de l'existence d'un ordre pour les éléments à fusionner.

4. Clusters : regroupement de triangles voisins issus d'un maillage.
5. Moyenne quadratique : c'est une moyenne d'une liste de valeurs, définie comme la racine de la moyenne des carrés des valeurs.

[Woo *et al.*, 2002] proposent une méthode qui fonctionne comme suit : le nuage de points est découpé en appliquant la méthode octree (voir figure 2.9). Cette solution consiste à diviser successivement le maillage initial en "cellules" (clusters) suivant un critère. Dans ces travaux, l'auteur utilise l'écart-type des valeurs de normale des points comme critère. Si cette valeur est supérieure à celle définie par l'utilisateur, la cellule initiale est divisée en huit cellules-filles en utilisant une décomposition octree. Le processus de division est stoppé lorsqu'une cellule satisfait les conditions définies, c'est-à-dire que la valeur de l'écart-type est plus petite que la valeur tolérée ou lorsque la cellule ne contient plus qu'une seule entité (triangle du maillage par exemple).

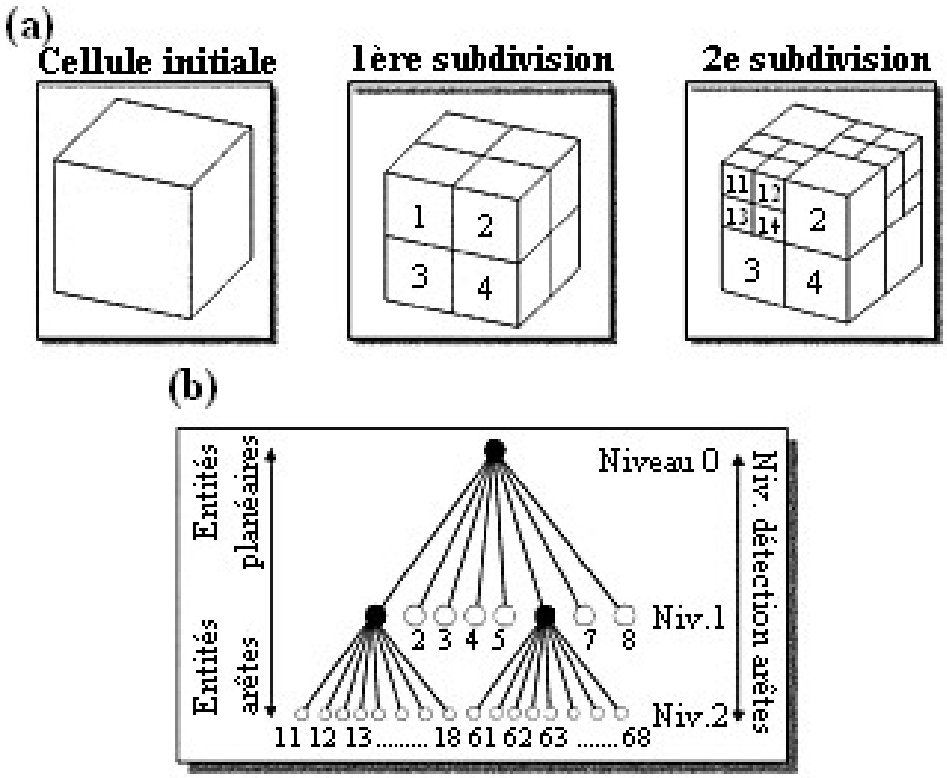


Figure 2.9 – Illustration de la méthode octree [Woo *et al.*, 2002] : (a) les divisions successives de cellules du cube et (b) le graphe octree correspondant.

Les travaux de [Attene *et al.*, 2006] sont assez similaires mis à part qu'ils utilisent un ensemble de primitives (plans, sphères et cylindres) et ils n'ont pas besoin de germe initial pour commencer la segmentation. En effet, ils partent du fait que chaque triangle constitue un unique cluster plat, ainsi le type "plan" est celui qui s'ajuste au mieux à chaque cluster. L'étape suivante consiste à agréger deux triangles voisins de manière hiérarchique. On attribue un coût à ces triangles, ce qui correspond à leur potentiel de fusion. Les régions sont alors fusionnées par ordre croissant de ces coûts. Ensuite, la méthode de [Attene *et al.*, 2006] fait appel à un graphe binaire (principe identique que celui de la figure 2.9 mais avec une division par 2). Leur critère de fusion est basé sur la distance minimale entre le cluster et la face (plan, cylindre, sphère) qui s'ajuste le mieux.

4. par clusterisation itérative : à la différence des méthodes précédentes, la clusterisation itérative est considérée comme paramétrique du fait que l'on connait le nombre de clusters final dès le début de la segmentation. Cette dernière peut être formulée sous la forme d'un problème de calcul variationnel. Ces approches utilisent l'algorithme k-moyennes (connu sous le nom de *k-means*[6]) comme dans les travaux de [Liu et Siegwart, 2013] où ils segmentent un nuage de points assez

6. K-means : méthode de clusterisation des données. Soit k une nombre de divisions donné, l'algorithme consiste à diviser l'ensemble des données en k clusters le tout en minimisant la somme des carrés des distances de chaque point par rapport à la moyenne des points.

bruité de l'intérieur d'un bâtiment.

5. par analyse spectrale : [Delest *et al.*, 2007] définit cette méthode par "un graphe G du maillage pouvant être représenté par une matrice d'adjacence. Cette matrice binaire $A(G)$ confronte tous les sommets du graphe et fournit $A_{ij} = 1$ s'il existe une arête reliant les sommets i et j. Le Laplacien L du graphe est une matrice associée qui correspond à $L = D - A$ où D est la matrice diagonale qui offre $D_{ii} = d$, la valence du i^e vertex". Les vecteurs propres du Laplacien jouent un rôle important dans le partitionnement du graphe. Une description plus précise est disponible dans les travaux de [Delest *et al.*, 2007, section 3.5].

6. par les méthodes implicites : comme le nom l'indique, elles font appel à des approches de partitionnement implicites. Le maillage est segmenté par rapport à une structure ou un objet tel qu'un squelette ou une image représentant le maillage. [Shamir, 2008] les classe en trois catégories : les méthodes basées sur la construction de contours, les approches *top-down* et celles qui s'appuient sur un squelette. Ces dernières sont les plus connues car on retrouve notamment les graphes de Reeb. Le découpage du nuage est effectué par des plans qui coupent le nuage à chaque nœud du graphe (cf. travaux de [Herlem *et al.*, 2012]).

7. par ligne de partage des eaux : cette dernière méthode consiste à considérer le maillage comme un relief avec des sommets positionnés à une certaine altitude. Le maillage est "rempli d'eau" et lorsque deux bassins se rencontrent, une ligne de partage des eaux est créée. Cette première méthode est appelée la *méthode ascendante* (figure 2.10 (a)) cependant il existe également une approche *descendante* (figure 2.10 (b)). Sur la première, on mesure le potentiel d'une frontière à rassembler deux régions alors que sur la deuxième, on s'intéresse à sa capacité à maintenir une séparation forte entre elles.

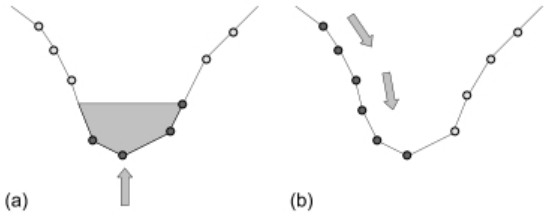


Figure 2.10 – Méthode de segmentation par ligne de partage des eaux : (a) l'approche ascendante et (b) l'approche descendante ([Delest *et al.*, 2007]).

[Chen *et al.*, 2009] ont réalisé un benchmark sur sept techniques. Ils présentent des illustrations des résultats ce qui permet de comparer aisément les bénéfices et les défauts de chaque technique. La base de données utilisée comporte 380 maillages répartis dans 19 catégories. La figure 2.11 est un extrait des résultats appliqué à des pièces mécaniques. On remarquera que la méthode qui segmente au mieux ces arbres est celle intitulée *Fit Primitive* (encadré rouge) qui correspond aux travaux de [Attene *et al.*, 2006]. En effet, les segments semblent correspondre à des entités métiers telles que les extrusions ou les poches. Les autres résultats donnent des segmentations assez aléatoires : chaque segment ne correspond pas à une entité de fabrication.

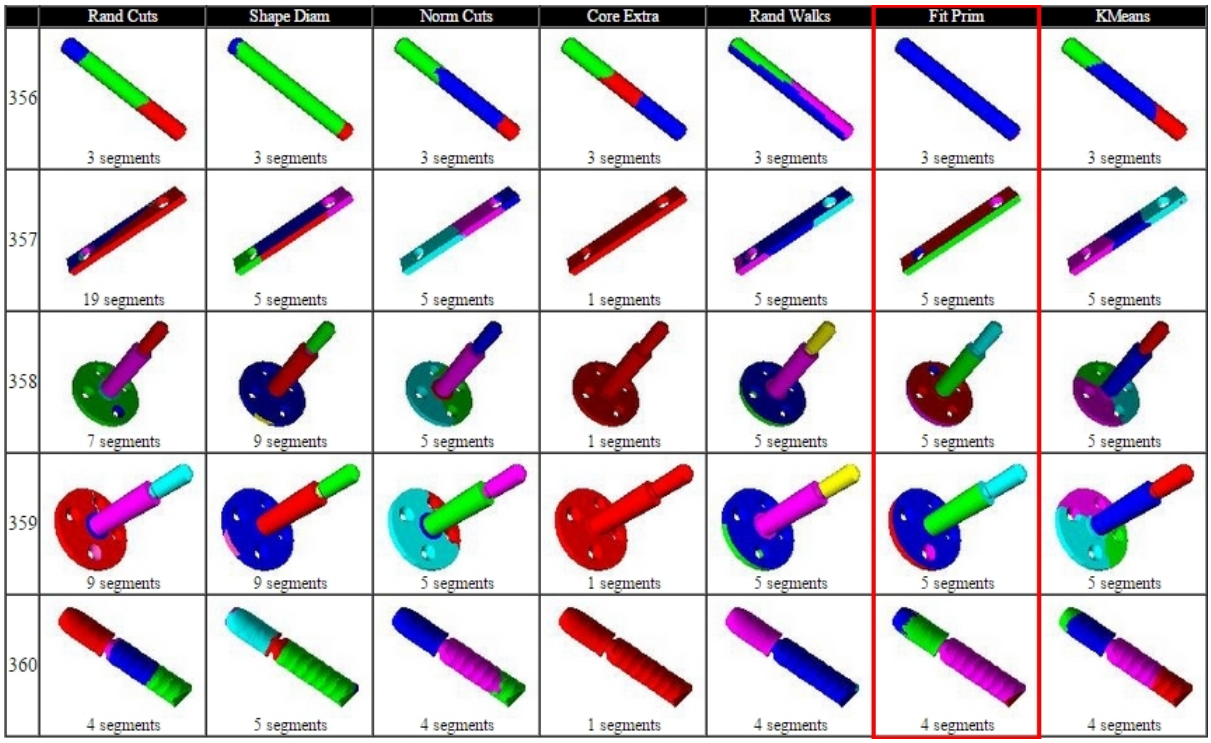


FIGURE 2.11 – Résultats des différentes techniques de segmentation sur un jeu de données issus des travaux de [Chen *et al.*, 2009].

2.1.2.4 Les techniques mixtes

Nous retrouvons une méthode segmentation hybride utilisant conjointement les arêtes et régions dans l'étude de [Sunil et Pande, 2008]. Ils considèrent autant les arêtes vives que les arêtes non-tranchantes (chanfrein ou arrondis par exemple) et classent les régions segmentées selon leur type et leur nature. La première basée sur les arêtes permet de classer les arêtes suivant leur "tranchant". Pour cela, des ratios ($\frac{H}{L}$ avec H la moyenne des hauteurs du triangle et L la moyenne des largeurs) sont calculés pour chaque triangle afin de déduire avec la fonction $tan\alpha$ la valeur des angles et déterminer si ces triangles du maillage sont obtus ou aigus. Les contours sont alors reconstitués par "croissance" d'arêtes et ceci de manière bi-directionnelle. Le processus est stoppé lorsque une boucle est fermée ou lorsqu'une arête déjà répertoriée est rencontrée ou s'il n'y a plus d'arête après le dernier sommet ayant satisfait le seuil du critère d'angle défini au préalable. La segmentation des régions est également réalisée par la technique de croissance à laquelle est associée des calculs de courbure de Gauss (K) et de courbure moyenne (H) entre chaque triangle et ses triangles voisins à fusionner. En fonction du signe des valeurs de K et H, le type de surface est déduit. On retrouve les équivalences entre les valeurs de courbure de Gauss et le type de face dans le tableau 2.3. L'avantage principal d'utiliser conjointement ces deux segmentations permet d'éviter que des régions s'entrecroisent lors de l'étape de croissance.

Type de surface	Courbure de Gauss (K)	Courbure moyenne (H)
Sphère convexe	+	-
Sphère concave	+	+
Cylindre convexe	0	-
Cylindre concave	0	+
Tore convexe	-	-
Tore concave	-	+
Plan	0	0

Tableau 2.3 – Équivalence entre les valeurs de K et H et le type de surface inspiré de [Sunil et Pande, 2008].

Dans ses travaux de thèse, [Chaperon, 2002] présente une méthode de segmentation d'un nuage de points très dense et bruité. Il scanne l'intérieur d'une usine et doit extraire de l'ensemble des points du nuage, un sous-nuage correspondant à une ligne de tuyauterie, en vue de récupérer le modèle 3D dans un modeleur CAO. Pour la segmentation, il part d'un point sélectionné par l'utilisateur et l'algorithme ajuste un cylindre idéal par rapport au point initial. Pour cela, il utilise les points voisins et il y applique un procédé de multi-résolution. Cette méthode consiste à ajuster différentes tailles de cylindres et à garder celle où le diamètre est suffisamment grand pour englober le maximum de points tout en évitant au mieux le bruit autour du cylindre (l'algorithme est décrit [Chaperon, 2002, page 161]). Une fois le cylindre idéal fixé, on peut alors propager cette méthodologie de chaque côté du cylindre.

Pour cette propagation, l'auteur commence par ajuster une demi-sphère à l'une des extrémités du cylindre afin de détecter un éventuel coude du tuyau. Puis il y ajuste un cylindre idéal et compare les deux résultats. La propagation s'arrête lorsque le nombre de points ajustés au cylindre idéal est trop faible.

[Xiao *et al.*, 2011] exposent une méthode permettant de segmenter des maillages issus de modèles CAO. La spécificité de leurs travaux réside dans le fait que ces maillages sont peu denses et les triangles sont assez longilignes. Leur proposition se décompose en trois étapes :

1. la classification des triangles du maillage en régions éparses et denses à l'aide de la méthode agglomérative et hiérarchisée de clusters proposée par [Attene *et al.*, 2006] ;
2. les régions éparses sont alors partitionnées en régions planes, cylindriques et coniques en faisant appel conjointement à la courbure de Gauss et à la transformée de Hough : l'algorithme est décrit dans [Xiao *et al.*, 2011, section 4] ;
3. les régions denses sont ensuite segmentées en appliquant un algorithme basé sur la courbure de Gauss.

La première étape qui consiste à séparer les régions denses des régions éparses en triangles permet d'effectuer une première segmentation, certes grossière, mais rapide comme on peut l'observer dans la figure 2.12. La segmentation utilisée dans la deuxième étape repose sur les techniques basées sur les arêtes.

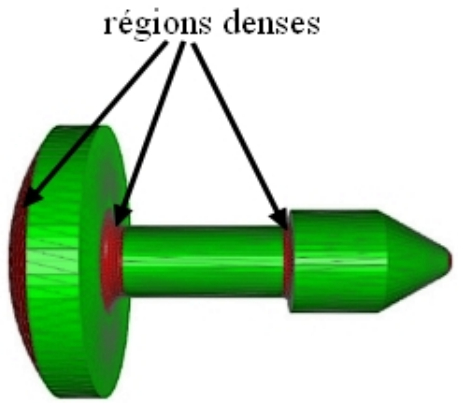


Figure 2.12 – Segmentation basée sur la densité de régions : en rouge les régions denses et en vert, les régions où le nombre de triangles est faible [Xiao *et al.*, 2011].

Conclusion sur la segmentation des données 3D

Les travaux de [Zhan *et al.*, 2010] avec les vecteurs normaux associés à des couleurs ne peuvent être retenus car ils traitent de scan unidirectionnel avec la détection de surfaces majoritairement planes, ce qui correspond au besoin de leur étude (détection de bâtiments dans le scan aérien d'un environnement). Les plans semblent être orientés dans une direction (scan vertical), ce qui dans notre cas, ne sera pas le cas. En effet, les données sont scannées dans plusieurs directions, puis les nuages de points sont assemblés. De plus dans l'étude [Zhan *et al.*, 2010], l'orientation des bâtiments dans le nuage de points est toujours identique d'un scan à l'autre. Dans notre cas, il faudrait alors toujours garder un repère fixe dans le nuage de points (exemple : un support, une table) ce qui est impossible à prévoir dans le cadre de nos travaux.

Quant à l'étude de [Fan *et al.*, 1987] avec la détection de frontières, il serait compliqué de l'appliquer à notre problème dans la mesure où elle permet de détecter les arêtes, ce qui ne nous donne pas un niveau d'information suffisant pour identifier un composant. Il serait cependant envisageable de la tester en complémentarité d'une technique par région. Elle permettrait peut-être de distinguer les limites entre composants d'un assemblage cependant, cette méthode a seulement été testée sur des images de profondeur (technique expliquée dans les travaux de [Chang et Park, 2001]).

Les travaux de [Attene *et al.*, 2006] sont prometteurs car l'utilisation de l'approximation de surfaces de type plan, cylindre, sphère permet ainsi d'obtenir des informations (type de segment) sur le segment dès l'étape de segmentation. De plus, la technique développée propose un gain de temps considérable par l'ordonnancement des éléments à fusionner. La figure 2.13 montre un exemple de segmentation réalisée par leur méthode sur une pièce mécanique. Le résultat de leur segmentation correspond à première vue aux faces du composant, ce qui concorde avec les éléments dont nous aurons besoin pour décrire la pièce (étape de signature - section 2.2). Les caractéristiques géométriques extraites de chaque segment permettront ensuite d'identifier la donnée.

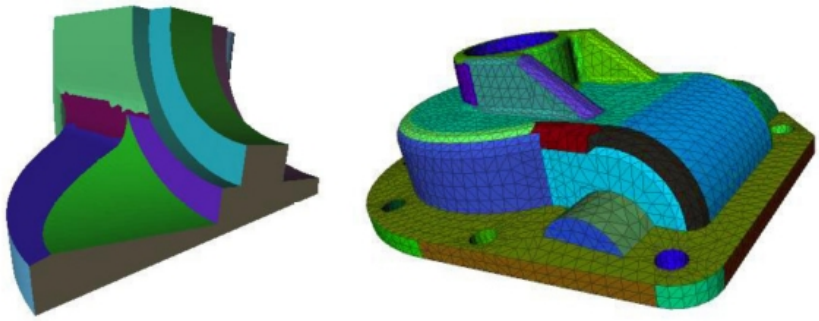


Figure 2.13 – Segmentation hiérarchique de deux modèles en régions ajustées par rapport à des plans, des cylindres et des sphères. La figure de gauche a été segmentée en 21 clusters et celle de droite en 45 clusters. [Attene *et al.*, 2006].

[Chaperon, 2002] propose une méthode robuste cependant son application est dédiée à des assemblages constitués de pièces cylindriques et standards. Cette solution serait applicable dans un contexte où l'objectif de l'activité de RE serait similaire (faisant appel à des éléments standards).

La première étape de la méthodologie de segmentation proposée par [Xiao *et al.*, 2011] pourrait être conservée afin de faire un tri dans les données, cependant cette méthode est testée sur des nuages de points générés à partir de données CAO. Il est possible que les valeurs de la densité des points dans les données issues de scan, ne soient pas aussi disparates.

Les travaux de [Sunil et Pande, 2008] sont similaires d'une part à ceux de [Xiao *et al.*, 2011] dans la mesure où ils font une distinction entre les triangles obtus et aigus. Cependant, ils permettent une segmentation plus précise de par le calcul du type de surface grâce à la courbure de Gauss et ce, malgré l'épaisseur fine de certaines des surfaces segmentées telles que nous pouvons l'observer dans la figure 2.14. D'autre part, leur segmentation aussi basée sur la croissance de région est prometteuse. Au final, il serait intéressant de tester leur technique afin de d'observer si cette distinction faite entre les triangles apportent un gain de temps ou pas.

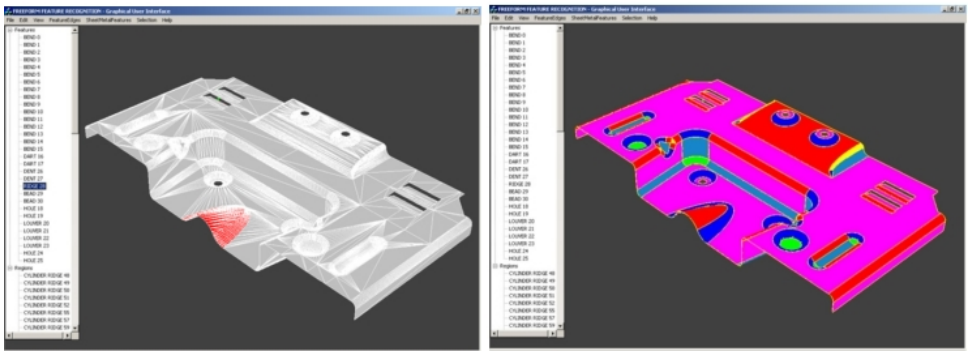


Figure 2.14 – Segmentation sur un STL issu de la CAO : à gauche la tôle initiale où l'on peut apercevoir les différents types de triangles et à droite la tôle après segmentation.

Une fois les données segmentées, nous allons maintenant étudier l'étape suivante : la signature de la donnée.

2.2 La signature des données

Après s'être intéressé aux différentes études sur la segmentation, c'est-à-dire aux moyens d'extraire les caractéristiques géométriques ou topologiques des données ; nous allons maintenant chercher à regrouper ces caractéristiques de sorte à créer des signatures. Ces signatures sont un moyen de formaliser les connaissances extraites des données.

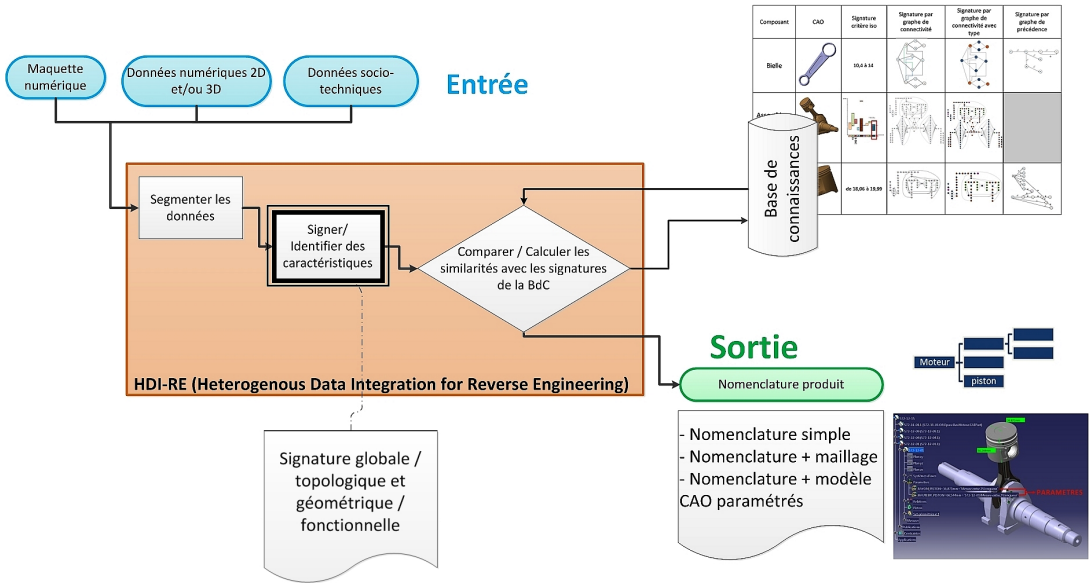


Figure 2.15 – Deuxième étape de HDI-RE : la signature des données

Pour cela, nous avons choisi de traiter simultanément les signatures 2D et 3D dans la mesure où les techniques de signature ne dépendent pas du type de données, mais plutôt du niveau de détails souhaité en sortie de cette étape (informations topologiques et géométriques sur la donnée d'entrée ou informations globales).

Cette étape de notre processus de RE (voir figure 1.17) est celle qui appartient au domaine du *Shape Matching* puisque c'est grâce à cette signature que nous pourrons comparer deux formes et/ou deux données.

Nous nous sommes appuyés des travaux de [Tangelder et Veltkamp, 2007] qui proposent de classifier les méthodes de *Shape Matching* en trois catégories :

- les techniques basées sur les caractéristiques ;
- les techniques basées sur les graphes ;
- les techniques basées sur la géométrie.

Ils proposent également une taxonomie que l'on peut retrouver dans la figure 2.16.

D'autres classifications existent comme celle proposée dans les travaux de [Coma *et al.*, 2003] où cinq catégories de techniques de reconnaissance de formes sont proposées. Il s'agit des techniques basées sur :

- la reconnaissance d'arêtes : chaque arête extraite de la donnée est analysée pour vérifier si elle est convexe, concave ou lisse.
- le volume et la décomposition convexe : il s'agit de soustraire le volume de l'enveloppe convexe du solide au solide lui-même. Le volume résultant est alors utilisé.
- la reconnaissance basée sur les graphes : cette technique est identique à celle proposée par [Tangelder et Veltkamp, 2007].
- la reconnaissance basée sur les réseaux de neurones : ces techniques sont majoritairement utilisées dans la reconnaissance de visages car elles ont la capacité d'apprendre par elles-mêmes grâce aux caractéristiques géométriques ou topologiques collectées sur un modèle initial.
- la reconnaissance basée sur des règles : c'est la méthode proposée dans les travaux de [Coma *et al.*, 2003]. Il s'agit de contrôler si les propriétés géométriques et la structure topologique du composant vérifient certaines règles définies.

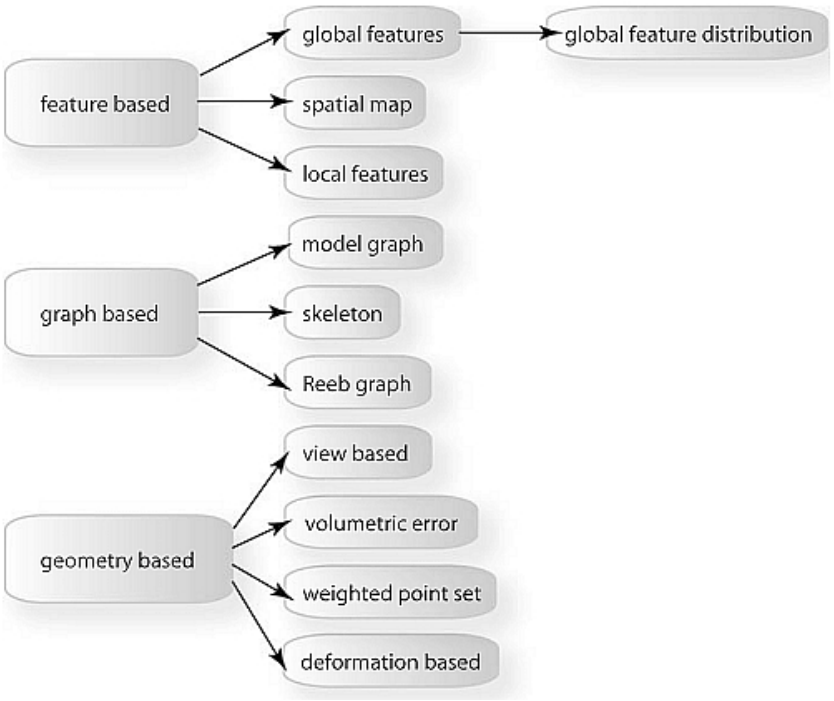


Figure 2.16 – Taxonomie des méthodes de *Shape Matching* par [Tangelder et Veltkamp, 2007].

Les travaux de [Tangelder et Veltkamp, 2007] servant de référence à un très grand nombre d'études dans le domaine, nous avons choisi de nous en inspirer afin de réaliser l'état de l'art sur les signatures. Notre signature peut être associée au terme *Shape* de *Shape Matching* qui se traduit par reconnaissance de forme. Le moyen de parvenir à cette reconnaissance réside, dans le cadre de nos travaux, à l'utilisation d'une signature de la donnée qui fournit une description de celle-ci. Quant au terme *Matching*, [Tangelder et Veltkamp, 2007] le définissent par "le processus permettant de déterminer la similarité entre deux formes". Nous ferons donc par la suite de la reconnaissance de signature, ce qui fera l'objet de la troisième étape de notre processus de RE (voir section 2.3 de l'état de l'art).

Parmi les travaux présentés dans les sections suivantes, certains présenteront des liens assez étroits avec la section concernant la segmentation. En effet, les travaux présentés précédemment traitaient de signature souvent par graphe, cependant le terme de signature n'était pas mis en valeur. D'où l'objet de cette section qui sera de mettre en lumière cette étape dans le processus de *Shape Matching*.

2.2.1 Les signatures basées sur les caractéristiques

D'après [Tangelder et Veltkamp, 2007], ces méthodes correspondent aux caractéristiques liées à une forme 2D ou 3D en utilisant un simple descripteur composé d'un vecteur de valeurs à d dimensions. La dimension d est fixée pour toutes les formes. La différence entre les caractéristiques globales et locales vient de l'objet qui est décrit. Si la description se reporte à une face par rapport à toutes les faces environnantes, alors on parlera de caractéristique locale. Dans le cas de caractéristiques décrivant l'ensemble des faces d'un modèle 3D, on les classera dans les caractéristiques globales. Ces dernières peuvent être un volume, une aire, un quotient de plusieurs caractéristiques. D'autres sont plus complexes, entre autres nous pouvons citer celui de compacité qui est fréquemment utilisé pour les données 3D et la circularité pour les données 2D.

[Montero et Bribiesca, 2009] proposent une étude bibliographique de ces deux caractéristiques globales permettant de décrire des formes 2D et 3D. Leur expression est basée sur une inégalité dite isopérimétrique remontant à l'époque Babylonienne qui utilise P le périmètre de la forme et A son aire.

$$P^2 \geqslant 4\pi A \tag{2.2}$$

De l'expression 2.2, ils en déduisent celle-ci :

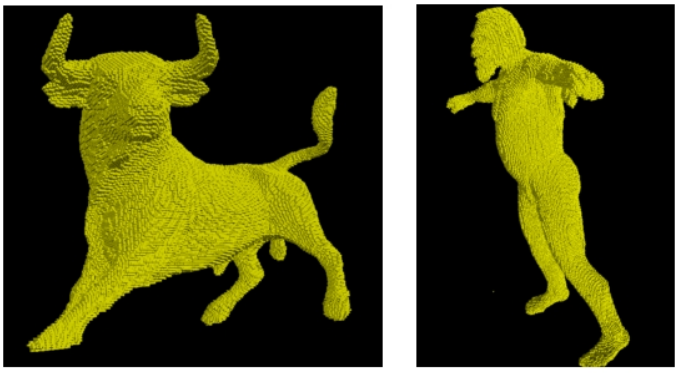
$$C = P^2/A \tag{2.3}$$

Dans cette deuxième formule 2.3, C correspond à la compacité d'une forme aussi appelée circularité, dispersion ou encore mesure de la rondeur qui appartiennent aux "facteurs de forme". Dans les espaces continus, nous retrouvons une version de l'inégalité 2.2 sous forme de volume :

$$A^3 \geqslant 36\pi V^2 \tag{2.4}$$

avec A la surface de la boîte englobante de la surface dans un espace 3D et V son volume.
La méthode qu'ils proposent repose sur l'application d'une compacité normalisée discrète nommée C_{DN} appliquée à des formes 3D et dont une description précise figure à la [Montero et Bribiesca, 2009, section 2.3]. Le concept repose sur l'utilisation de voxels [7] sur lesquels on calcule l'aire de leurs faces.

Leur critère prend également en compte la disposition des voxels les uns par rapport aux autres (surfaces en contact), ce qui permet d'obtenir des valeurs de compacité faibles. Dans la figure 2.17, la valeur de compacité C est de l'ordre des milliers. Leur méthode peut être également associée aux méthodes basées sur les caractéristiques locales de part l'utilisation de ces voxels qui sont des entités indivisibles de la donnée 3D.



(a) 3D Bull shape; surface area=64670, volume=357664, C_{DN}=0.9752, C=2114.2587

(b) 3D Man shape; surface area=65984, volume=274828, C_{DN}=0.9619, C=3803.59

Figure 2.17 – Deux différentes formes et leur mesure de compacité par [Montero et Bribiesca, 2009] : C la compacité classique et C_{DN} la compacité proposée dans l'étude.

[Osada *et al.*, 2002] étudient une fonction mesurant les propriétés globales d'un objet. Cette fonction est basée sur la distribution des distances et se veut discriminante par rapport aux autres fonctions testées. Leur technique appelée *D2 Shape Distribution* mesure la distance entre deux points sur la surface, choisis de manière aléatoire. La figure 2.18 montre un extrait des résultats présentés dans ces travaux. Nous observons que l'allure des courbes varie en fonction de l'objet. [Osada *et al.*, 2002] ont développé un moteur de recherche en ligne afin de fournir aux utilisateurs un outil permettant de récupérer des modèles 3D polygonaux signés par rapport à leurs attributs de forme.

7. Voxel : c'est un néologisme qui est la contraction de *volume* et *pixel* tout comme pixel l'est pour *picture* et *element*. Ce type de représentation permet de stocker différents types d'informations comme les coordonnées spatiales du pixel, la matière de l'élément, une taille relative par rapport aux autres voxels etc.

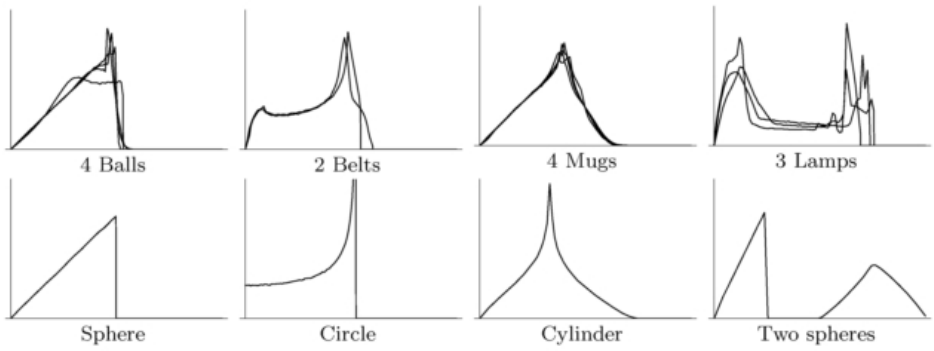


Figure 2.18 – Signature globale par distribution de distances sur différents modèles 3D d'après [Osada *et al.*, 2002], test réalisé sur 133 modèles groupés dans 25 classes.

Parmi les autres signatures basées sur les caractéristiques, nous retrouvons les méthodes basées sur les cartes spatiales. Ces dernières fonctionnent avec en entrée les positions physiques de l'objet ou de ses régions qui sont arrangées de sorte à ce que leur position relative dans l'objet soit respectée. Cependant ces méthodes dépendent principalement du repère de l'objet et de son orientation ; les rotations de ce dernier affectant alors les cartes spatiales. [Ankerst *et al.*, 1999] appliquent cette méthode afin de réaliser une classification des molécules dans le domaine de la chimie. Ils découpent la molécule selon des "coquilles" concentriques (*shell*) et par secteurs dans lesquels ils calculent ensuite des distances quadratiques entre chaque sous-élément. Ces distances sont ensuite représentées par un histogramme comme nous pouvons l'observer dans la figure 2.19. Dans ces travaux, la molécule est signée seulement dans une vue, ce qui ne pourrait être possible dans le cas d'opérations de scan routinier car le nuage de point de la donnée ne serait pas orienté à l'identique d'une opération à l'autre.

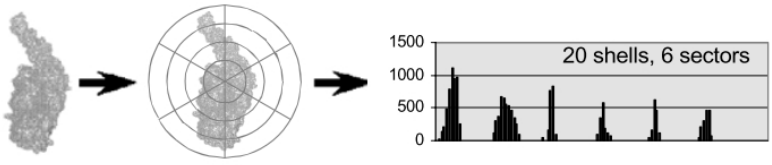


Figure 2.19 – Représentation d'une molécule par une carte spatiale et un histogramme des formes (d'après [Ankerst *et al.*, 1999]).

2.2.2 Les signatures basées sur les graphes

Par rapport aux signatures précédentes qui prennent en compte uniquement la géométrie de la pièce pour en extraire une caractéristique, les méthodes de signatures basées sur les graphes ont pour but de donner un sens aux informations géométriques extraites afin de décrire comment les formes des objets sont liées entre elles. [Tangelder et Veltkamp, 2007] proposent de les classer en trois catégories :

1. **les modèles graphiques** : on retrouve les graphes non-orientés qui permettent de décrire un modèle 3D (un B-Rep par exemple) où les nœuds et arêtes du graphe correspondent respectivement aux faces du B-Rep et aux liens entre ses faces. C'est ce que [El-Mehalawi et Miller, 2003a] appellent les graphes étiquetés. Ils associent aux arêtes du graphe autant d'informations que possible concernant les surfaces et les arêtes de l'objet. Le graphe est représentatif de la topologie quant aux étiquettes [8], aussi appelée attributs des arêtes, elles représentent la géométrie du composant. Les informations pour la construction du graphe sont extraites du STEP du modèle CAO. Sur l'image de gauche de la figure 2.20, nous retrouvons le modèle 3D et à droite sa représentation

8. Un graphe étiqueté est un graphe dont les arêtes ont reçu des attributs. Il peut s'agir de poids (valeur). On parle alors de graphe pondéré.

sous forme de graphe. Le but de leurs travaux est de trouver un moyen de comparer facilement et rapidement deux modèles CAO afin de retrouver des composants dans une base de données.

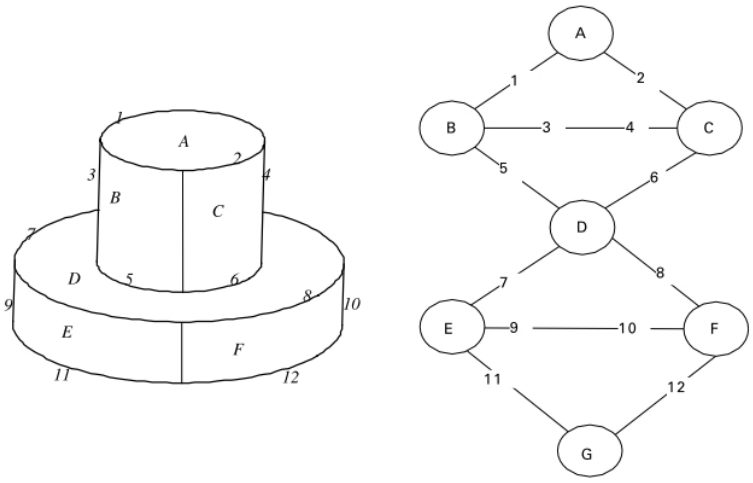


Figure 2.20 – Une pièce CAO et sa signature par graphe étiqueté (d'après [El-Mehalawi et Miller, 2003a].

2. **les squelettes** : [Sundar *et al.*, 2003] les définissent comme une "colonne vertébrale" de l'objet. Pour les données 2D telles que les images, cela s'apparente à l'axe médian de l'image alors que pour les données 3D, une "surface" médiane est calculée. Par rapport aux graphes classiques, les squelettes sont plus simples (le nombre d'éléments est inférieur à celui d'un graphe). La construction du graphe est réalisée en deux phases : l'une consiste à affiner l'objet pour obtenir des points et l'autre à segmenter l'objet en tronçons. L'opération d'affinage consiste à pondérer chaque point du squelette par une valeur (distance euclidienne entre chaque voxel et l'enveloppe englobante du tronçon par exemple). Tous les points sont ensuite reliés. Les proportions de chaque tronçon du squelette restent proportionnelles par rapport à chaque segment de l'objet grâce à l'application de l'algorithme d'arbre couvrant de poids minimal (appelé *Minimum Spanning Tree* en Anglais) [9]. Le graphe orienté correspondant est créé en commençant par les points ayant la valeur la plus grande jusqu'à la plus faible. La figure 2.21 illustre le type de graphe obtenu par [Sundar *et al.*, 2003] pour différents types d'objets.

Figure 2.21 – Signature par graphes de squelette d'après [Sundar *et al.*, 2003].

3. **les graphes de Reeb** : ce dernier type de graphe est similaire au précédent. Ce qui le diffère c'est la manière de construire le squelette et le critère de pondération. [Tierny *et al.*, 2006] proposent une méthode de construction de graphe de Reeb en trois étapes : l'extraction de sommets caractéristiques, le calcul d'une fonction d'application et la construction de lignes de niveau discrètes. Le principe du graphe de [Reeb, 1946] est de découper un élément par des lignes de

9. L'arbre couvrant de poids minimal d'un graphe est un sous-ensemble du graphe permettant de connecter tous les sommets ensemble et dont la somme des poids des arêtes est minimale ; le poids de l'arête correspondant à sa longueur.

niveaux parallèles (voir figure 2.22). Dans chaque tronçon obtenu, le centre de gravité est calculé, correspondant à un nœud du graphe. Une fois tous les nœuds reliés, le graphe est créé. La fonction f permettant de créer ces lignes de niveaux a fait l'objet d'un très grand nombre d'études. Quant aux travaux de [Tierny *et al.*, 2006], ils se veulent novateurs dans la mesure où ils présentent des propriétés d'invariance et de forte tolérance aux variations de position des modèles et de résolution des maillages. Nous pouvons également citer les travaux de [Tung et Schmitt, 2005] qui ont proposé une méthode de graphes de Reeb améliorés leur permettant d'indexer des données dans une base de données.

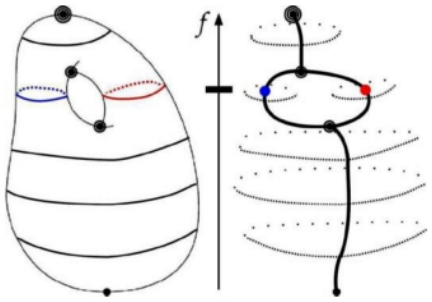


Figure 2.22 – Application du principe du graphe de Reeb à une pièce simple avec f la fonction permettant de créer les lignes de niveaux.

2.2.3 Les signatures basées sur la géométrie

[Tangelder et Veltkamp, 2007] répertorient plusieurs catégories concernant les signatures qui s'appuient sur la géométrie. Ils proposent ainsi quatre techniques basées sur :

1. **les vues** : l'idée principale de ces méthodes repose sur le fait que deux modèles 3D sont similaires s'ils se ressemblent sous tous leurs angles de vue. Le but de ces méthodes est de représenter un objet 3D avec un ensemble de vues 2D (images binaires ou des esquisses) ce qui réduit le problème à une comparaison en 2D au lieu de 3D. [Cyr et Kimia, 2003] ont recours à une méthode basée sur l'apparence qui repose sur la similarité d'une image d'intensité [10] projetée parmi les vues voisines. On applique pour cela un descripteur afin de déterminer la direction principale de leur variation ce qui permet ensuite de les comparer. Le plus utilisé dans la littérature est le *Principal Component Analysis* (PCA). De plus, afin de minimiser leur nombre, les vues sont segmentées : chaque segment représentant une vue. L'ensemble des vues est représenté grâce à un "graphe d'aspect", où certaines vues sont jugées comme similaires (selon un critère de similarité défini). La figure 2.23 illustre d'une part une représentation avec des vues classiques (à gauche) et d'autre part une représentation avec des vues d'aspect (à droite).

   Selon [Petre *et al.*, 2010], les approches basées sur les vues dépendent de plusieurs paramètres : le nombre de projections, la position de l'appareil photo qui assure une optimale description de l'objet (c'est-à-dire les vues), le choix du descripteur de forme 2D et l'algorithme employé pour la phase de comparaison. Les vues sont généralement obtenues par silhouettes qui sont l'ombre propre [11] de l'objet. C'est le cas des travaux de [Yang *et al.*, 2008] dont un exemple est présenté dans la figure 2.24 avec un modèle de chaise. On distingue 10 silhouettes de l'objet auxquelles est appliqué un descripteur de champ lumineux (calcul de moment par exemple) permettant de caractériser chaque vue.

10. Image d'intensité : cette technique repose, en partie, sur la réflectivité de l'objet touché par une impulsion laser. L'intensité est une mesure, collectée pour chaque point, de la force de retour de l'impulsion laser ayant généré le point. Les valeurs sont ensuite retraitées pour reconstituer les images (souvent utilisées dans l'imagerie aérienne).
11. L'ombre propre apparaît sur l'objet quand le volume de l'objet se soustrait aux rayons incidents d'une source lumineuse.

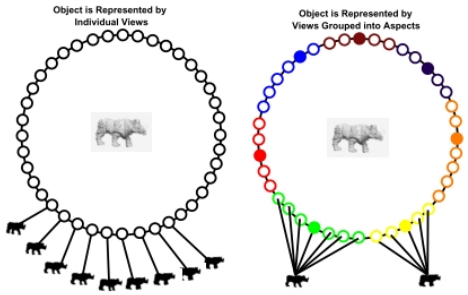


FIGURE 2.23 – Représentation d'un modèle 3D par un graphe d'aspect des vues 2D de l'objet : à gauche, toutes les vues obtenues après rotation de l'objet par rapport au sol et à droite, certaines vues, jugées similaires (même couleur) sont représentées par une vue unique [Cyr et Kimia, 2003], réduisant ainsi leur nombre.

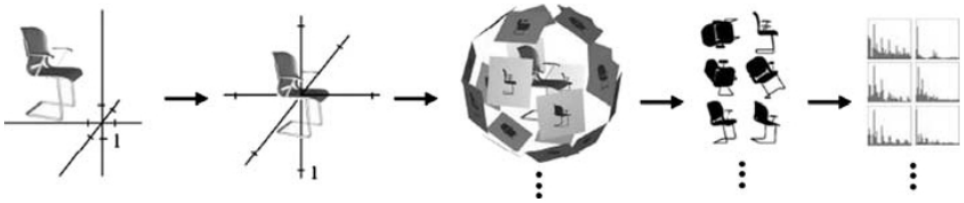


FIGURE 2.24 – Signature basée sur la géométrie et utilisant l'ombre propre de l'objet [Yang *et al.*, 2008].

2. **les erreurs volumétriques** : le principe de ces signatures consiste à calculer l'erreur volumétrique entre un objet et une séquence d'enveloppes extérieures associées à un autre objet. [Sánchez-Cruz et Bribiesca, 2003] présentent une méthode basée sur l'erreur volumétrique entre deux formes voxélisées ; la figure 2.25 illustrant cette méthode. Ils calculent l'effort, suivant la quantité de voxels et la distance, qu'il faudrait fournir pour déplacer les voxels d'une forme à l'autre. Cependant leur méthode s'applique uniquement sur des objets invariants (position, volume, échelle) afin que la superposition d'une paire d'objets soit toujours la même. Ainsi si l'un des objets venait à être déplacé (changement d'angle de position), la méthode ne pourrait être appliquée.

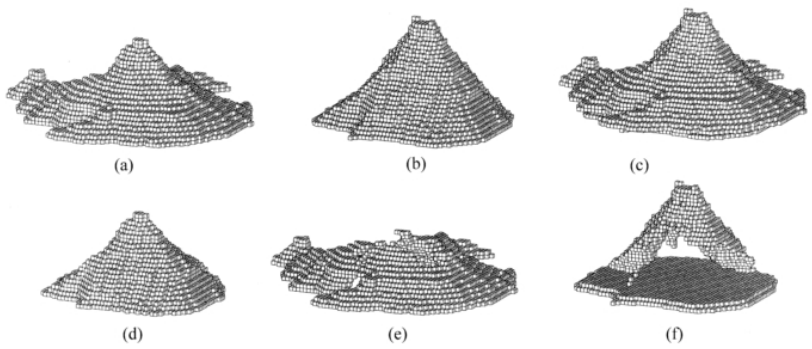


FIGURE 2.25 – Voxelisation d'objets afin de déterminer leur dissemblance : (a) l'objet O_1, (b) l'objet O_2, (c) superposition des deux objets, (d) voxels communs aux deux objets, (e) voxels à déplacer et (f) voxels négatifs où les voxels positifs seront placés (d'après [Sánchez-Cruz et Bribiesca, 2003]).

3. **les points sélectionnés et pondérés** : il s'agit de représenter la forme 3D comme un ensemble de points pondérés. [Tangelder et Veltkamp, 2003] enferment chaque objet dans une grille 3D et génèrent une signature représentant l'objet par un ensemble de points pondérés. Ils testent trois types de signature. La première dans laquelle ils prennent le sommet de la grille avec la valeur de courbure de Gauss la plus élevée et lui assignent comme poids cette valeur. Pour la deuxième, ils

observent la variation de la normale pour chaque cellule de la grille 3D à laquelle ils attribuent la valeur. La dernière méthode, appelée par "points-moyens", consiste à ajouter à chaque cellule de la grille le centre de gravité des sommets de la cellule avec un poids unitaire.

La méthode proposée par [Funkhouser *et al.*, 2004] consiste à pondérer les points par la somme des distances entre chaque modèle et plus exactement à partir de chaque point de l'une des surfaces jusqu'au point le plus proche de l'autre surface et vice-versa. Cette méthode ressemble à celles où il est question calculer l'effort à fournir pour déplacer les points d'un objet à l'autre (cf. travaux de [Sánchez-Cruz et Bribiesca, 2003]).

4. **la déformation** aussi appelée évolution de la forme : ce type de signature s'applique principalement aux données 2D. Il consiste à comparer une paire de formes 2D en mesurant la déformation nécessaire pour les superposer. [Basri *et al.*, 1998] utilisent une fonction de coût qui pèse la similarité de points homologues sur deux courbes (voir Figure 2.26) par rapport aux propriétés locales des points telle que la distance entre eux ou la distance avec la tangente ou encore la courbure du contour de ces points. La similarité globale des contours est alors définie par la somme des coûts locaux.

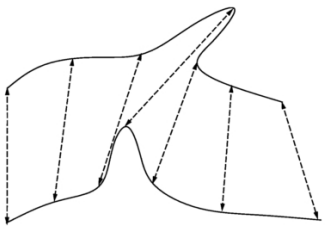


Figure 2.26 – Signature par mesure de la déformation des contours d'une forme 2D par rapport à une autre par [Basri *et al.*, 1998]. Les points entre les deux contours sont mis en correspondance (hypothèses de départ). Le coût est calculé pour chaque point repéré.

Cette méthode est également similaire aux deux autres présentées précédemment à la différence que le problème est réduit dans un espace à deux dimensions.

Conclusion sur les méthodes de signature

En fonction de l'objectif souhaité par l'utilisateur et du type de données disponible en entrée du processus de Reverse Engineering, un ou plusieurs types de signatures pourront être employés. Pour identifier une pièce dans un assemblage, les méthodologies par graphe semblent les mieux adaptées. Il serait intéressant de tester cette solution. En effet, elles permettent de décrire un objet autant topologiquement que géométriquement. Les graphes de [Reeb, 1946] sont certes très utilisés pour les données de type maillage cependant d'après [Tangelder et Veltkamp, 2007], elles présentent plusieurs inconvénients. Elles sont tout d'abord sensibles aux changements topologiques et aux données bruitées (faces manquantes par exemple), ce qui perturbe le calcul des différentes distances.

Dans le cas de Reverse Engineering permettant de comparer une nomenclature à une nomenclature ou bien le maillage d'un assemblage à celui d'un autre, les signatures par caractéristique globale ou par géométrie pourraient répondre à notre besoin. Les méthodes par caractéristique globale permettent de donner un résultat d'ensemble sur la donnée cependant aucune caractéristique géométrique n'est conservée. Elles simplifient les problèmes de comparaison de forme puisque la comparaison s'effectue au niveau de la caractéristique. De plus, ce type de signature propose un gain de temps considérable pour l'ensemble du processus de RE (notamment pour la signature puis la comparaison). Il permet également de réaliser une classification entre des objets de différentes familles (balle, tasse, animaux etc.) cependant la distinction entre deux éléments de la même famille reste encore un sujet à étudier.

Les signatures basées sur la géométrie sont un peu moins adaptées à notre problème car pour la plupart (travaux de [Cyr et Kimia, 2003; Sánchez-Cruz et Bribiesca, 2003]), elles dépendent de l'orientation

de la pièce lors de la signature. Dans notre cas, nous ne pourrions pas identifier la présence d'un composant à l'intérieur d'un assemblage. De plus dans les travaux de [Basri *et al.*, 1998; Tangelder et Veltkamp, 2003], le niveau d'information fourni par leur solution n'est pas suffisant pour aider à la reconstruction d'une maquette numérique avec des modèles CAO ou même des enveloppes externes. Comparées aux méthodes par caractéristique globale, la question du délai de traitement se pose car nous pouvons imaginer qu'il pourrait être plus long. En effet, il faut considérer le temps de création des vues ou des voxels ou de la grille 3D.

Pour conclure, nous garderons les signatures par caractéristique globale lorsque nous souhaiterons obtenir un niveau d'information global (pour la comparaison des nomenclatures des maquettes numériques par exemple). Puis nous étudierons les signatures par graphe pour récupérer des informations géométriques et topologiques afin de reconstruire des modèles CAO paramétrés (ou pour obtenir les enveloppes externes des modèles).

2.3 Le Knowledge Based Engineering vers la comparaison des signatures

D'après [Tangelder et Veltkamp, 2007], l'étape de "Matching" est le processus permettant de déterminer la similarité entre deux formes et c'est souvent réalisé en calculant la distance entre elles. Dans la majorité des études présentées précédemment, l'étape de signature et celle de comparaison des signatures sont très liées au point qu'elles ne soient parfois, pas tout à fait distinctes l'une par rapport à l'autre. C'est le cas par exemple, dans les travaux de [Basri *et al.*, 1998] où l'étape de comparaison empiète sur celle de signature.

Dans le cadre de nos travaux, nous considérons l'existence d'une base de connaissances servant à effectuer la comparaison de signatures. Nous nous intéressons alors à la pertinence des données qu'elle pourrait contenir afin d'identifier les données acquises. Puis dans un deuxième temps, nous verrons comment comparer les signatures entre elles. Cette deuxième partie sera divisée en deux sous-parties traitant des techniques de comparaison pour les signatures par graphe et par géométrie.

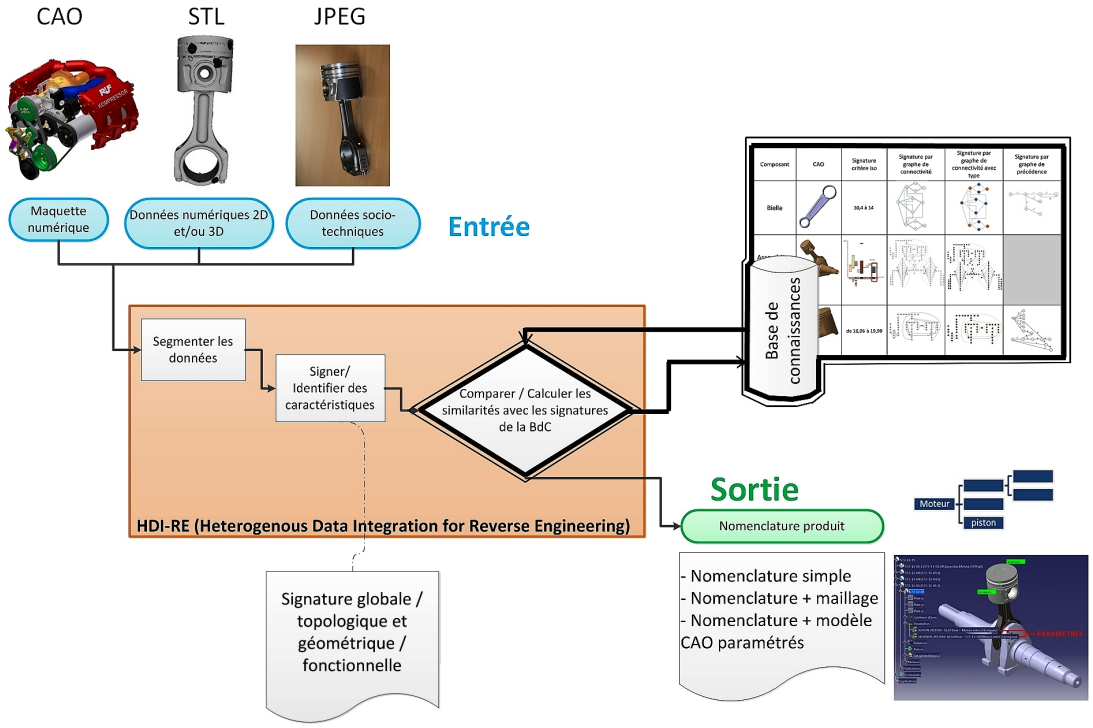


Figure 2.27 – Troisième étape de HDI-RE : la comparaison des signatures.

Ainsi, cette section se décomposera en deux parties, tout d'abord nous étudierons les travaux dans le domaine du KBE afin de voir le type de connaissances qui pourrait nous être utile de stocker dans le cadre de RE routinier ainsi que l'utilisation de ces connaissances. Ensuite nous verrons les études qui traitent de la comparaison des signatures.

2.3.1 Le Knowledge-Based Engineering (KBE)

[Chapman et Pinfold, 1999] définissent le KBE comme une méthode d'ingénierie qui combine la programmation orientée objet, les techniques d'intelligence artificielle et les outils de CAO. Les outils de KBE permettent de capturer les informations liées au produit et de les traiter de manière à les réutiliser pour les activités d'ingénierie. Ensuite l'utilisation d'un modèle permet d'automatiser en partie ou intégralement le processus. Le but ultime des systèmes de KBE est de capitaliser les meilleures pratiques de conception ainsi que les informations d'expertise métiers en les stockant dans une base de connaissances propre à une entreprise.

[Oldham *et al.*, 1998] apportent une méthodologie nommée MOKA (pour *Methodology and tools Oriented to Knowledge-based engineering Applications*) et considèrent l'existence d'un cycle de vie pour les systèmes de KBE composé de huit étapes qui sont chronologiquement : l'identification, la justification, la capture, la formalisation, le paquetage, la distribution, l'introduction et enfin l'utilisation. Les trois que nous retiendrons sont celles de capture, de formalisation et de paquetage. Celle de capture consiste à collecter les connaissances à partir des personnes ou à partir d'un répertoire de connaissances. Puis elles sont structurées et représentées sous une forme informelle (diagrammes le plus souvent). Pour l'étape de formalisation, il s'agit de récupérer la connaissance structurée précédemment et de la représenter dans un format cohérent, formel et neutre afin que puisse être évaluée son exactitude et sa capacité à être ré-utilisée. Le répertoire de connaissances peut alors être mis à jour avec les nouvelles connaissances et sa cohérence est vérifiée. La dernière étape de paquetage (*package* en anglais), il s'agit de récupérer les connaissances dans le répertoire et de les traduire, manuellement ou automatiquement, sous le format correspondant au système de KBE destinataire. Différents tests sont à nouveau réalisés afin de s'assurer que l'efficacité et la robustesse du système n'ont pas été corrompus lors de mauvaise traduction ou de changement imprudent.

Les travaux de [Thompson *et al.*, 1999] présentent un prototype logiciel appelé REFAB (Reverse Engineering FeAtured-Based) permettant de faire du RE tout en utilisant les caractéristiques de fabrication comme primitives géométriques. Ces caractéristiques peuvent être : un simple trou, un trou taraudé, une rainure simple ou profilée, une poche rectangulaire ou profilée, un bossage ou encore un chanfrein arrondi ou profilé, etc. L'utilisation de celles-ci permet d'augmenter la précision et la facilité d'exploitation de la démarche de RE. La reconnaissance des caractéristiques est semi-automatique car l'utilisateur doit d'abord sélectionner quel type de caractéristique est visible. Puis il sélectionne sur une des vues sa position approximative. Le logiciel détermine ensuite les paramètres liés à la caractéristique par rapport à la position des données 3D.

[Lovett *et al.*, 2000] étudient la faisabilité de l'application des systèmes KBE à des entreprises de petite et moyenne tailles (PME). Ils comparent notamment MOKA à d'autres systèmes tel que CommonKADS^{TM} qui est utilisé plus généralement dans les Knowledge-Based Systems (KBS). Leurs travaux visent à appliquer les grands principes du KBE à une méthode qui serait implémentée dans des PME anglaises. Selon eux, une application de KBE possède trois composantes :

- la géométrie, qui est un élément substantiel de la CAO. La plupart des logiciels de KBE ont la capacité de faire de la CAO.
- la configuration, qui consiste à trouver une combinaison valide de composants.
- les connaissances métiers, qui prennent en compte la fabrication et autres contraintes qui servent à concevoir le produit.

Ces composantes peuvent ensuite être pilotées par des règles tels que des formules mathématiques, des états conditionnels etc. Le principal avantage de l'approche de KBE est la réduction des délais à travers l'automatisation des tâches routinières.

[Fisher, 2004] montre un moyen de reconstruire des modèles à partir de données scannées incomplètes où une partie de l'objet n'a pu être scannée (objet qui en masque un autre). Pour cela, il fait appel à un algorithme qui planifie les différentes vues de l'objet afin d'anticiper les parties dissimulées. Cette solution s'applique uniquement aux surfaces planes et cylindriques. Il projette les surfaces dans les zones occultées tant qu'une partie de la surface est observable (voir figure 2.28). Il part de l'hypothèse que la partie occultée est similaire à celle qui est visible et que les connaissances acquises sur la deuxième sont applicables à la zone à reconstruire. Il utilise cette méthode lorsque les nuages de points sont incomplets ou très bruités.

Figure 2.28 – Reconstruction de surfaces à partir de nuages de points incomplets [Fisher, 2004] : à gauche, deux cylindres occultés par une surface et à droite, les cylindres reconstruits.

Il est important de citer le projet de recherche AIM@SHAPE (**A**dvanced and **I**nnovative **M**odels **A**nd **T**ools for the development of **S**emantic-based systems for **H**andling, **A**cquiring, and **P**rocessing knowledge **E**mbedded in multidimensional digital objects) mené de 2004 à 2007. Le but de ces travaux de recherche est de formaliser les connaissances encapsulées dans la représentation numérique d'objet. Pour cela, trois niveaux de représentation sont considérés :

- la géométrie qui permet de visualiser la forme ;
- la structure qui indique comment les composants de l'objet sont liés entre eux (tel un squelette)
- la sémantique qui propose une interprétation de l'objet numérisé.

Les résultats des recherches constituent une base de connaissances très riche et disponible en ligne [Visionair, 2015].

Les travaux de [De Luca *et al.*, 2006] dans le domaine de l'architecture sont intéressants dans la mesure où ils font appel à deux types de données : les images et les nuages de points. Les images donnent des informations sur la texture des surfaces et sont calibrées et mises à l'échelle grâce au nuage de points. Un autre aspect de ces travaux concerne la présence d'une base de connaissances qui sert à capitaliser les informations extraites. Cette base contient également des signatures sémantiques permettant de caractériser des éléments géométriques extraits des données. L'auteur classifie ainsi les différents éléments d'architecture pour les colonnes corinthiennes (voir figure 2.29).

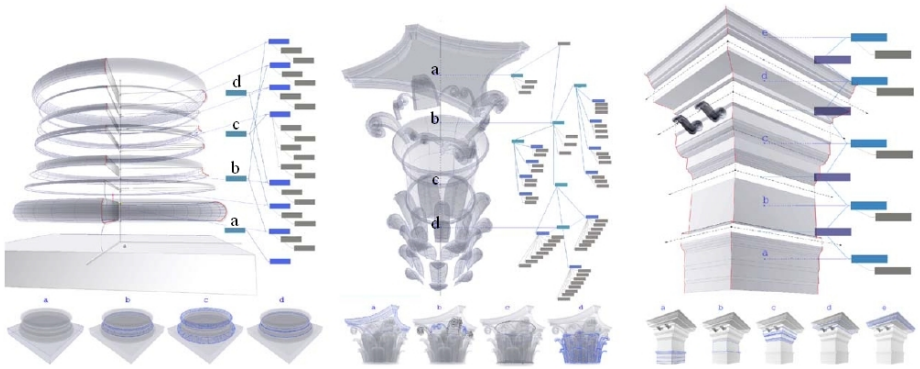


Figure 2.29 – Représentation sémantique d'éléments architecturaux : a. la base, b. le chapiteau c. l'entablement etc. ([De Luca *et al.*, 2006]).

[Nüchter et Hertzberg, 2008] amènent une solution afin d'interpréter automatiquement les surfaces segmentées d'un nuage de points d'une scène intérieure. Pour cela, ils utilisent un modèle générique basé sur des connaissances simples en bâtiment. Ce modèle (nommé "réseau de contraintes") est composé de différents nœuds reliés entre eux par des relations géométriques. On retrouve un exemple de ce modèle dans la figure 2.30 où l'on peut apercevoir des différents relations pour une porte (*door*) : elle est orthogonale au sol (*floor*) ainsi qu'au plafond (*ceiling*) et elle n'est pas parallèle au mur (*wall*).

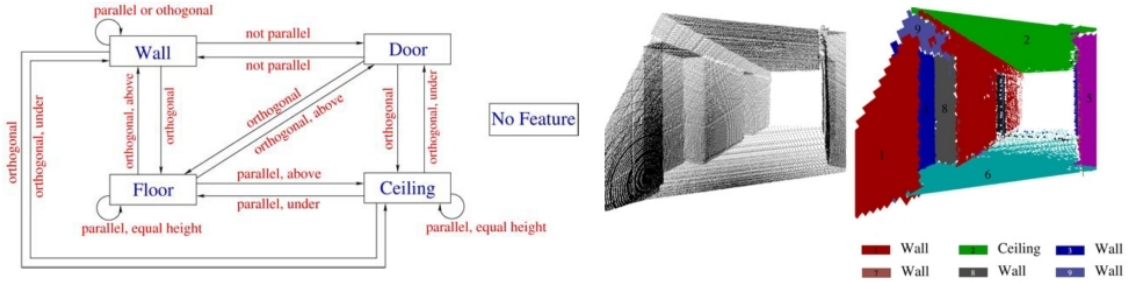


Figure 2.30 – Interprétation d'une scène intérieure scannée à partir d'un modèle générique d'après [Nüchter et Hertzberg, 2008].

Le nuage de points est segmenté en plans grâce à la méthodologie RANSAC ([Fischler et Bolles, 1981]) puis un algorithme vérifie toutes les contraintes entre les plans afin d'interpréter ensuite chaque composant de la scène : mur (en rouge), sol (en bleu clair), plafond (en vert), porte (en bleu foncé).

Les travaux de [Durupt *et al.*, 2009] proposent une méthode RE qui intègre une phase d'extraction des connaissances, permettant de lister les primitives géométriques, aussi appelées caractéristiques standards. Ils en définissent deux types : celles de conception (caractéristiques donnant une fonction) et celles de fabrication (caractéristiques qui permettent de fabriquer la pièce selon un processus, comme les dépouilles pour le moulage). Ils emploient un squelette structurel et fonctionnel permettant de paramétrer les caractéristiques standards et de piloter leur géométrie. Ce squelette, appartenant à une base de connaissances, est en 3D et sous forme filaire (rendu en 2D). Sa méthode, nommée Knowledge-Based for Reverse Engineering (KBRE), permet de reconstruire ainsi des cylindres, des cônes, des parallélépipèdes et des volumes pyramidaux. La figure 2.31 montre un squelette dont les différentes caractéristiques sont paramétrées en fonction des valeurs extraites du nuage de points. L'arbre de construction de la pièce et les fonctions liées à chaque entité (poche, extrusion etc.) sont reconstruits pas à pas. Un des buts de cette méthodologie est aussi de capitaliser les intentions de conception [12] du produit afin de guider la démarche de RE.

12. Intentions de conception : aussi appelées "informations d'expertise métiers" (caractéristiques standards). Elles correspondent aux méthodes de conception (extrusion, poche etc.) mais aussi aux procédés de fabrication.

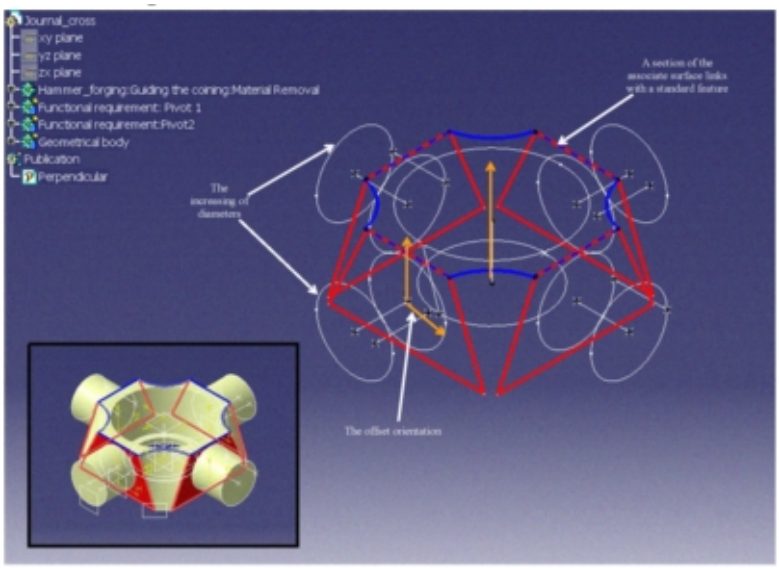


Figure 2.31 – Squelette paramétré d'un croisillon de Peugeot 403 dans CATIA V5 par [Durupt *et al.*, 2009]

[Amadori *et al.*, 2012] font appel aux méthodes de KBE afin de créer un outil de génération automatique de modèles CAO à partir de *templates*[13] avec une géométrie de haut niveau. Leur solution repose sur la génération de modèles standards qui représentent une classe d'objets. Les connaissances sont stockées sous forme de règles de conception, de relations et de données pour chaque *template*. [Amadori *et al.*, 2012] proposent de distinguer deux types de transformations géométriques permettant d'instancier un composant. Il s'agit des transformations morphologiques et topologiques. Les transformations morphologiques impliquent une transformation de la forme des objets. Pour ce qui est des topologiques, il s'agit de transformations qui impliquent la localisation des caractéristiques ou des objets dans un repère géométrique. Il existe trois types de transformations topologiques :

- en ajoutant une instance : un objet est placé à un endroit spécifique ;
- en enlevant une instance : un objet est retiré d'une position choisie ;
- en remplaçant une instance : un objet est enlevé et un autre, appartenant à une autre classe, est ajouté.

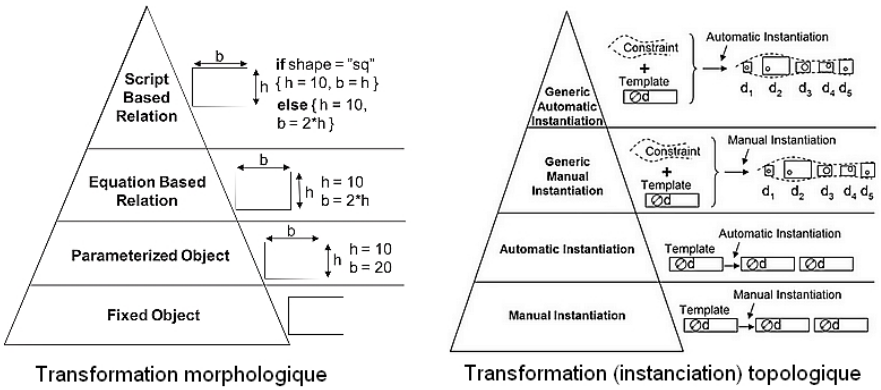


Figure 2.32 – Différentes étapes des transformations morphologiques et topologiques sur un modèle CAO [Amadori *et al.*, 2012].

13. Template : squelette paramétré d'un modèle CAO qui devient un modèle CAO à part entière dès lors qu'on affecte des valeurs à ses paramètres. On parle alors d'une instanciation.

La figure 2.32 illustre ces deux notions et permet de voir les différents niveaux de connaissances utilisés en fonction de la complexité de l'étape de modélisation. Par exemple pour des transformations morphologiques, les relations basées sur un script (*Script Based Relations*) correspondent au niveau le plus élevé. Il s'agit de créer des relations en utilisant un langage de programmation fourni par le logiciel CAO avec des règles qui régissent ces relations. Concernant les transformations topologiques, elles se définissent selon trois items : le *template*, les contraintes et le contexte. Pour le premier item, il s'agit du modèle initial à re-instancier. Concernant les contraintes, ce sont les conditions à satisfaire par les instances à instancier. Pour le dernier (le contexte), il s'agit de l'environnement géométrique de l'instance à laquelle les contraintes se réfèrent. De la même manière que pour les transformations morphologiques, elles sont classées par niveau dont le plus élevé (*Generic Automatic Instanciation*) est l'étape où les fonctions pré-définies peuvent automatiquement générer ou supprimer des instances en fonction du choix de l'utilisateur. Les instances sont contextuellement dépendantes et capables de varier de manière paramétrée, par conséquent leur réutilisation complète et l'automatisation de l'instanciation sont réalisées.

Leur solution, nommée HLCt (pour *High Level CAD template*) permet de créer par exemple un modèle CAO d'avion à partir d'un *template* paramétré. L'utilisateur renseigne les différentes valeurs et choisit les configurations (type d'aile, de propulsion, nombre de portes, de sièges etc.) puis le système crée automatiquement le modèle d'avion correspondant. Pendant l'étape d'instanciation, un moteur d'inférence [14] localise la géométrie stockée dans la base de connaissances pour ensuite, lui affecter les paramètres. Il vérifie également l'ajustement de tous les sous-assemblages les uns par rapport aux autres. Une fois toutes les conditions remplies, le modèle CAO est construit.

Récemment les travaux de [Ali, 2015] traitent d'une méthodologie appelée REFM (pour Reverse Engineering For Manufacturing) associant les connaissances métiers liées à la fabrication du produit (étapes d'usinage), au processus de RE. Pour cela, elle utilise un graphe de précédence des surfaces usinées [15] qui sert à identifier chaque composant de la base de données. Ce graphe est utilisé comme *roadmap* afin d'identifier les surfaces segmentées issues du nuage de points en entrée du système. La figure 2.33 résume la méthodologie REFM. Cette dernière se décompose en deux étapes : l'une de capitalisation qui permet d'apporter une démarche structurée à l'utilisateur dans son processus de RE en vue de re-fabriquer la pièce (choix du matériau, étapes de fabrication etc.). Les connaissances sont capitalisées. Les données en sortie de cette étape permettent ensuite de générer la gamme de fabrication selon l'APEF (Avant-Projet d'Étude de Fabrication). La deuxième étape concerne la ré-utilisation des connaissances acquises au préalable, en réutilisant les gammes de fabrication de pièces similaires déjà rétro-conçues par le système.

14. Moteur d'inférence : c'est un outil utilisé en intelligence artificielle permettant aux systèmes experts telles que les méthodes de KBE de conduire des raisonnements logiques et de déduire des conclusions à partir d'une base de faits et d'une base de connaissances (d'après [Wikipédia, 2015]).

15. C'est un graphe orienté dont les nœuds, représentant les surfaces usinées, sont ordonnés en fonction des étapes d'usinage définies par l'utilisateur.

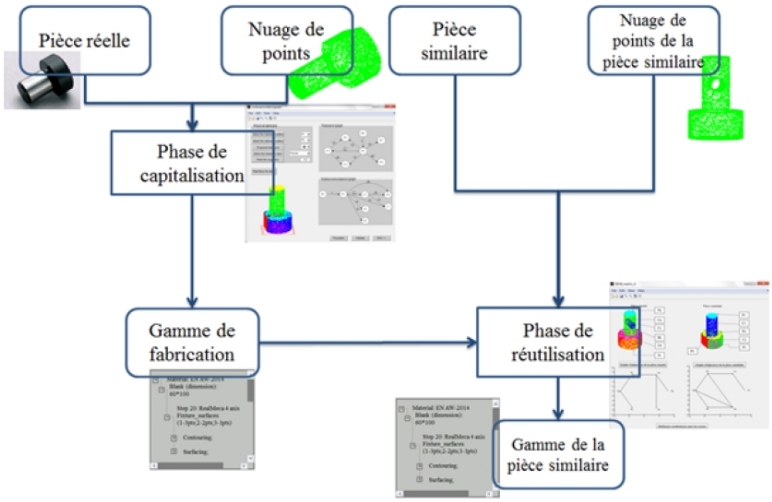


Figure 2.33 – Méthodologie REFM basée sur une phase de capitalisation puis de ré-utilisation des connaissances [Ali, 2015].

Conclusion sur le KBE

Tout d'abord, les résultats de [Fisher, 2004] pourraient être réutilisés dans nos travaux car ils abordent le problème de complétude des données cependant seuls les cylindres et les plans sont traités. Ensuite le logiciel REFAB de [Thompson *et al.*, 1999] est limité à des pièces seules cependant il est intéressant de garder en mémoire la réutilisation des caractéristiques de fabrication en vue de reconnaître semi-automatiquement des caractéristiques et reconstruire ainsi un modèle 3D de la pièce.

La méthodologie MOKA proposée par [Oldham *et al.*, 1998] conforte le fait d'associer des méthodologies KBE pour répondre à notre problématique. En effet, MOKA présente des similitudes avec notre proposition. La capture ou l'extraction des informations puis leur formalisation s'apparente à nos étapes de segmentation et de signature.

Ensuite, la solution proposée par [Nüchter et Hertzberg, 2008] pourrait nous intéresser car même si elle est appliquée au domaine du bâtiment (scène avec des plans positionnés géométriquement les uns par rapport aux autres), nous pourrions imaginer utiliser le même principe afin de reconnaître différents composants dans un assemblage. Il reste cependant à traiter le positionnement des pièces qui risque de varier d'un scan à l'autre.

Quant aux travaux de [Ali, 2015] et [Durupt *et al.*, 2009], ils découlent de problématiques très proches de la nôtre et il semble évident que nos travaux s'appuieront sur leurs résultats malgré que ces deux études ne traitent pas d'assemblages. Enfin l'étude de [Amadori *et al.*, 2012] nous conforte dans l'utilisation de *templates* dans notre démarche de RE. En effet, selon leur niveau de détails, les *templates* permettront de reconstruire la maquette numérique jusqu'aux modèles CAO paramétrés.

Tous ces travaux nous ont montré l'importance de la présence d'une base de connaissances afin de guider notre démarche de RE. En effet, le Knwoledge-Based Engineering, apporte une autre dimension au *Shape Matching*. Les solutions apportées par la reconnaissance de forme seule ne permettent pas de reconstruire une maquette numérique avec une niveau de détails suffisant pour répondre à notre problématique. Concernant le problème de complétude de données, nous pouvons imaginer reconnaître même partiellement une donnée et ainsi la traiter, voire reconstruire son modèle 3D, ceci grâce à la présence de la base.

2.3.2 La comparaison des signatures (ou calculs des similarités)

Cette dernière étape de notre processus concerne la comparaison de la signature de notre donnée d'entrée avec la base de connaissances contenant des signatures. Elle est très importante car elle conclut la démarche de RE, tout comme dans le domaine du *Shape Matching* où une forme est reconnue lorsque nous avons pu l'identifier. Ici l'identification repose sur la comparaison de deux signatures que l'on appelle aussi calcul des similarités (ou des distances). Ainsi nous identifierons la présence d'un composant dans un assemblage en retrouvant sa signature dans les données d'entrée. Nous traiterons de la comparaison de graphe et de la comparaison géométrique. Pour ce qui concerne, les signatures globales, les comparer reviendrait à comparer des nombres. Dans le cadre de nos recherches, cette comparaison peut se faire très simplement, sans faire appel à aucun algorithme. C'est la raison pour laquelle nous choisirons de ne pas traiter ce type de comparaison.

2.3.2.1 La comparaison des signatures par graphe

Le domaine de la comparaison de graphe est très vaste et complexe. Nous partons du principe que les lecteurs de ce manuscrit ne sont pas des spécialistes dans ce domaine-là. Cette section propose ainsi un aperçu des travaux existants dans le but d'en comprendre rapidement les concepts et ceci afin de pouvoir comparer différentes solutions par la suite.

Les travaux de thèse de [Bärecke, 2009] nous permettent de comprendre le concept de similarité ou d'égalité de graphe. Il distingue deux catégories : l'égalité exacte et l'égalité inexacte.

Pour ce qui est de l'égalité exacte, il s'agit d'une égalité structurelle entre deux graphes. Ceci est formalisé par le problème d'isomorphisme exact de graphes. C'est le cas le plus connu qui est utilisé en particulier dans le domaine de la reconnaissance de formes. Ce type de comparaison revient à répondre à la question d'égalité entre deux graphes par OUI ou NON. L'avantage de cette approche est le fait qu'elle soit structurée, ce qui permet d'obtenir une comparaison plus précise par rapport à une approche globale. En effet, dans le domaine du Shape Matching, on cherche à identifier une forme à partir de sa structure. Par exemple, si l'on souhaite reconnaître une bielle, on peut utiliser l'information qu'elle est composée de deux cylindres reliés par un parallélépipède. Ensuite lorsqu'on cherche à savoir si un graphe est inclus dans un autre graphe, alors on parle d'isomorphisme de sous-graphes. Le but de l'algorithme est donc de trouver le plus grand sous-graphe pour lequel il y a égalité. C'est ce qu'on appelle dans la littérature le *maximum common subgraph* (MCS). La figure 2.34 illustre l'isomorphisme de sous-graphes.

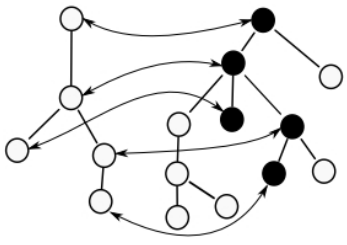


Figure 2.34 – Isomorphisme entre un graphe et un sous-graphe (en noir).

Concernant les égalités inexactes (deuxième catégorie), il s'agit ici de calculer la distance entre les deux graphes car les graphes ne possèdent pas par exemple, la même structure ou possèdent des variations des attributs. Le problème est formalisé par le calcul de la minimisation de distance. Cette approche est employée dans le traitement d'images afin de comparer par exemple deux images de la même scène prises à deux moments distincts ou lorsque les données sont bruitées. En considérant que les différences entre les graphes sont petites, on parle alors d'isomorphisme inexact de graphes ou de sous-graphes. Les algorithmes basés sur cette approche ne garantissent pas de trouver la meilleure solution mais ils fournissent au moins une bonne approximation de la solution et dans un temps raisonnable.

[Wilson et Zhu, 2008] proposent une étude comparative des méthodes spectrales. Cette approche consiste à transformer le problème à celui du calcul des valeurs propres et fonctions propres d'une matrice, c'est ce qu'on appelle déterminer le spectre de la matrice en question. En général, on s'intéresse à la matrice d'adjacence ou à la matrice Laplacienne normalisée. L'auteur décrit dans ses travaux les représentations standards utilisées pour les graphes et compare la cospectralité des graphes. Il s'agit de voir si deux graphes ont les mêmes valeurs propres par rapport à la représentation matricielle utilisée.

[Bärecke, 2009] présente également un autre type d'approche appelée heuristique. Selon lui, la plupart des heuristiques ne garantissent pas une solution optimale mais elles recherchent une solution de bonne qualité. Dans le cas de problèmes complexes, elles permettent de réduire considérablement le temps de calcul.

[Conte *et al.*, 2004] proposent un état de l'art assez exhaustif de toutes ces approches depuis les trente dernières années. Ils présentent une taxonomie de tous les algorithmes utilisés dans le domaine du *Pattern Recognition* (équivalent au *Shape Matching*).

Grâce aux travaux de [Conte *et al.*, 2004; Bärecke, 2009; Wilson et Zhu, 2008], nous proposons un récapitulatif de toutes les méthodes de comparaison de graphes existantes dans le tableau 2.4.

Egalité exacte	Egalité inexacte	Autres Techniques
Recherche d'arbre [Conte, 2004] : • Constrainst Satisfaction Problem (CSP) • Algorithme d'Ullmannn • Netgraph • Algorithme VF (ou VF2)	Recherche d'arbre [Bärecke, 2009] [Conte, 2004] : • Error-correcting algorithm • Algorithme de relaxation discrète	Heuristique et méta-heuristique [Bärecke, 2009] • algorithmes gloutons • algorithmes évolutionnaires (EA) • par approche génétique
Autres méthodes [Conte, 2004] : • Algorithme Nauty • Algorithme RETE	Optimisation continue [Conte, 2004] : • Relaxation labelling algorithm • Expectation-Maximization Algorithm • Weighted Graph Matching Problem (WGM) • Graduated Assignment Graph Matching (GAGM) • Fuzzy Graph Matching (FGM) • Reproducing Kernel Hilbert Space	Elastic Graph Matching (EGM) [Conte, 2004] Bunch Graph [Conte, 2004]
	Méthodes spectrales [Wilson, 2008] : • Adjacency matrix • Combinatorial Laplacian matrix • Normalized Laplacian matrix • Heat Kernel • Path Lenght Distribution	
	Autres méthodes [Bärecke, 2009] [Conte, 2004] : • par réseaux de neurones • l'optimisation par colonies de fourmis • par le tabou réactif • par décomposition	

Tableau 2.4 – Classification des techniques de comparaison de graphes.

Nous allons présenter différents travaux illustrant quelques unes des approches répertoriées dans ce tableau.

[El-Mehalawi et Miller, 2003b] proposent de comparer les graphes associés à des pièces mécaniques présents dans une base de données. Ils utilisent pour cela une méthode basée sur l'égalité inexacte et par recherche d'arbre. Ils débutent par une indexation des graphes dans la base de données ce qui permet de réduire le nombre de graphes à comparer (on garde les graphes correspondant à des pièces similaires). Cette indexation est réalisée par l'ajout de différents paramètres à chaque graphe :

— N_t : le nombre total de nœuds du graphe ;
— P : le nombre de surfaces de type plan ;
— C_1 : le nombre de surfaces cylindriques ;
— C_2 : le nombre de surfaces coniques ;
— S : le nombre de surfaces sphériques ;
— T : le nombre de surfaces toriques.

L'indexation est accomplie en se basant sur les caractéristiques des graphes et suivant l'ordre suivant :

- le nombre de nœuds moins le nombre de surfaces toriques : $N = N_t - T$;
- le nombre de nœuds correspondant à des plans : P ;
- le nombre de nœuds correspondant à des cylindres ou des cônes : $C = C_1 + C_2$.

Les graphes mis de côté sont ensuite comparés à l'aide d'un algorithme basé sur l'égalité inexacte. Pour deux graphes G_i et G_j, les distances suivantes sont calculées : $d(N_i, N_j) < s$ pour le nombre de nœuds, $d(P_i, P_j) < s$ pour le nombre de plans et $d(C_i, C_j) < s$ pour le nombre de cylindres et cônes avec s un seuil (entre 0 et 1) défini par l'utilisateur et d la distance définie par l'équation suivante :

$$d(a, b) = \frac{|a - b|}{max(a, b)} \tag{2.5}$$

Une fois la comparaison entre toutes les arêtes du graphes effectuée, un critère de similarité est calculé. Sa formule est la suivante :

$$Si = \frac{2N_S}{N_0 + N_j} \frac{E_1}{E_2} \tag{2.6}$$

avec N_S le nombre de paires de nœuds similaires, N_0 le nombre de nœuds du graphe que l'on compare, N_j le nombre de nœuds du graphe candidat (issu de la base de données), E_1 le nombre d'arêtes compatibles et qui sont connectées à des nœuds compatibles et E_2 le nombre total d'arêtes connectées aux nœuds compatibles.

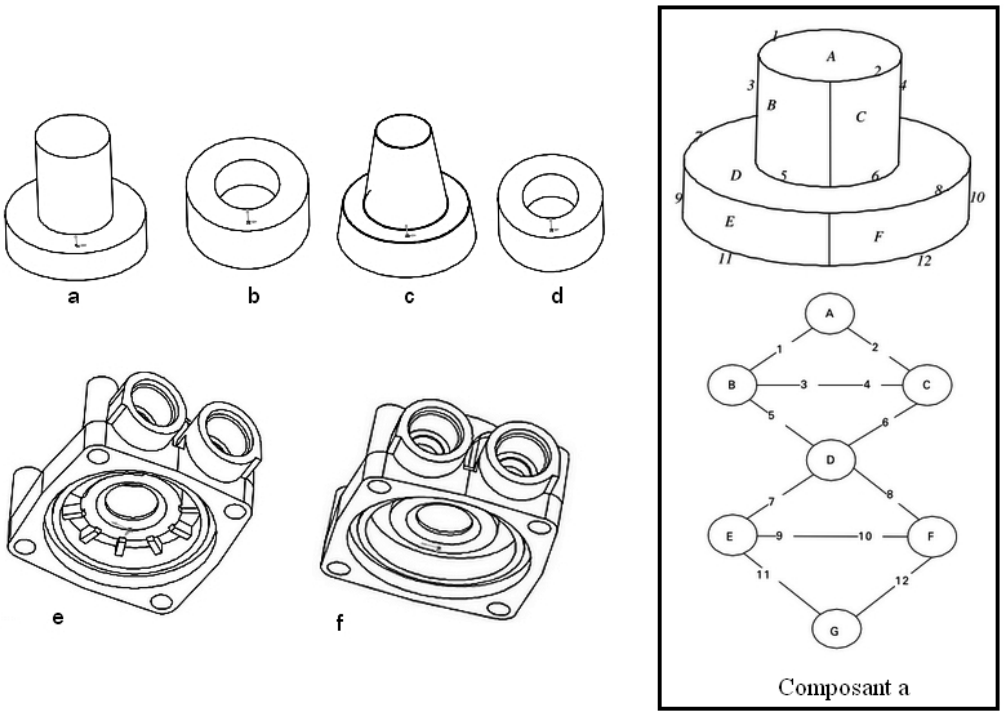


Figure 2.35 – Ensemble de données comparées dans les travaux de [El-Mehalawi et Miller, 2003b].

Un ensemble de composants testés dans l'étude est présenté dans la figure 2.35. Nous pouvons observer le graphe du composant a dans l'encadré de la figure. En appliquant leur algorithme sur les formes cylindres a et b, ils obtiennent un nombre de paires de nœuds compatibles de 4. Le nombre de nœuds de chaque composant comparé est de 7. Le nombre d'arêtes qui appartiennent aux nœuds égaux est de 12 (dans cet exemple, c'est le cas des deux pièces). Le facteur de similarité est alors de :

$$Si = \frac{2 \times 4}{7 + 7} \times \frac{12}{12} = 57\%.$$

Les travaux de [El-Mehalawi et Miller, 2003b] sont à considérer car leur méthode semble rapide et efficace concernant la comparaison topologique. De plus, leur solution utilisant l'indexation afin de réduire

le nombre de comparaison de graphes, conviendrait à nos travaux. En effet, la comparaison des pièces mécaniques e et f donne une similarité de 86 % pour des graphes d'environ 200 nœuds. Cependant leur algorithme possède quelques lacunes. La comparaison des cylindres a et c donne un critère de similarité de 100% alors que ce sont deux pièces différentes.

2.3.2.2 La comparaison des signatures géométriques

Ce type de comparaison de signature est à proprement parlé de la reconnaissance de forme. Il s'agit en particulier du domaine communément appelé *Pattern Recognition* étant donnée que la signature est associée à une forme géométrique.

[Veltkamp et Hagedoorn, 2001] proposent un état de l'art, cité à de nombreuses reprises et sur lequel nous nous appuierons afin d'exposer les principales méthodes existantes. Le tableau 2.5 résume les différentes techniques en fonction du type de données : pour les points qui sont comparés un à un ou plusieurs par plusieurs, de même pour les courbes puis les régions.

Ensemble de points	Courbe	Régions
Distance de Hausdorff	Distance de Hausdorff	Distance de Hausdorff
Par "bottleneck"	Distance de Fréchet	Distance de Fréchet
Par le poids minimum	Fonction par angle cumulatif	Fonction par angle cumulatif
Par uniformité (ou équilibre)	Fonction par signature	Par recouvrement d'aire et différence symétrique
Par la déviation minimale	Par longueur d'arc affine	
Par subdivision d'une transformation de l'espace	Par réflexion métrique	

Tableau 2.5 – Classification des techniques de comparaison pour les signatures par géométrie.

Pour commencer, nous allons expliquer en quoi consiste la distance de Hausdorff. Il s'agit de calculer des distances minimales entre deux polygones A et B. Pour cela, il est facile d'imaginer que la distance la plus courte entre ces deux polygones correspond à la distance d'un point appartenant à A le plus proche de l'un des points appartenant à B. Cependant cette distance minimale ne tient pas compte de la forme et de la position des polygones. La distance de Hausdorff considère justement ces formes ainsi que leur position. Elle se définit par la distance maximum d'un ensemble de points jusqu'au point le plus proche de l'autre ensemble. Sa formule est la suivante :

$$h(A,B) = max_{a\in A}\{min_{b\in B}\{d(a,b)\}\} \tag{2.7}$$

avec a et b des points appartenant respectivement aux ensembles A et B et $d(a,b)$ correspondant à la distance euclidienne entre chacun de ces points. Ensuite, la distance de Hausdorff est sensible à la position des polygones. Prenons l'exemple de la figure 2.36. Nous avons deux polygones P_1 et P_2 avec deux cercles ayant pour rayon $H(P_1,P_2)$. Nous observons alors que la valeur de la distance de Hausdorff augmente ou diminue en fonction de la position des polygones.

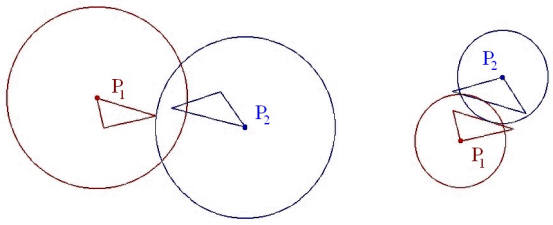


Figure 2.36 – Distance de Hausdorff en fonction de la variation de la position de deux polygones.

Si nous calculions la distance minimale entre ces deux polygones de manière simple (telle qu'imaginée précédemment), alors ce serait les deux polygones de gauche qui auraient une valeur de distance la plus

petite. Or dans le cas de la distance de Hausdorff, la valeur de la distance des deux polygones de gauche est la plus grande.

Pour la comparaison de lignes, la distance de Hausdorff s'applique à chaque point définissant cette ligne. La figure 2.37 illustre l'application de la distance de Hausdorff à deux images. Le but est de retrouver le petit avion en haut à gauche dans l'image d'ensemble. Un filtre de Canny est d'abord appliqué afin d'extraire les contours des formes dans chaque image. Les polylignes correspondantes à l'image de l'avion (petite image) sont ensuite comparées aux polylignes de l'image d'ensemble (figure b). Après avoir lancé l'algorithme de Rucklidge qui minimise la distance de Hausdorff, nous pouvons apercevoir le meilleur résultat de cette comparaison (contours en rouge dans la figure c.)

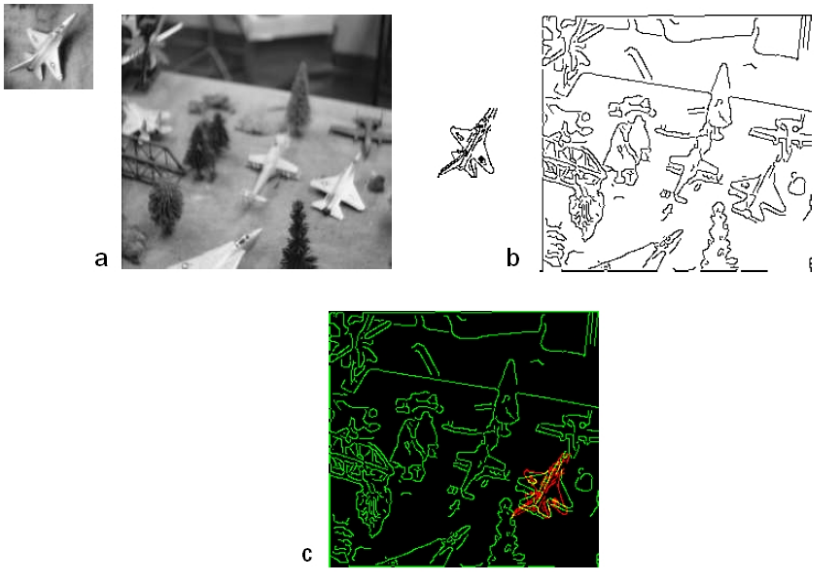


Figure 2.37 – Application de la distance de Hausdorff entre deux images d'après [Leventon, 2015] : (a) images à comparer ; (b) contours des images ; (c) matching entre les deux images après avoir appliqué la distance de Hausdorff.

Un deuxième type de comparaison concerne la distance de Fréchet utilisée pour la comparaison de courbes ou de régions. Il s'agit de calculer la similarité entre deux courbes par exemple en prenant en compte leur position et l'ordre des points le long de ces courbes. La figure 2.38 illustre la différence entre distances de Hausdorff et de Fréchet.

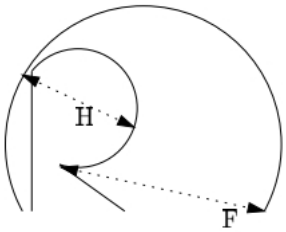


Figure 2.38 – Distance entre deux courbes avec Hausdorff (H) et Fréchet (F) d'après [Veltkamp et Hagedoorn, 2001].

La distance de Hausdorff n'est pas appropriée parfois pour calculer la différence entre deux courbes. Pour Fréchet, il faut imaginer marcher le long de ces courbes A et B. Pour tous les points de A, la distance jusqu'au point le plus près de B peut être petite mais si on se déplace simultanément sur les deux courbes et que l'on mesure la distance entre les points correspondants, le maximum de ces distances peut être

plus grand. C'est ce que l'on appelle la distance de Fréchet. De manière formelle, on considère A et B, les courbes paramétrées $A(\alpha(t))$ et $B(\beta(t))$ avec leur paramétrisation α et β des fonctions continues de même paramètre $t \in [0, 1]$ telles que $\alpha(0) = \beta(0) = 0$ et $\alpha(1) = \beta(1) = 1$.

La distance de Fréchet est le minimum sur tous les accroissements monotones des paramétrisations $A(\alpha(t))$ et $B(\beta(t))$ de la distance maximale :

$$d(A(\alpha(t)), B(\beta(t))), t \in [0, 1]. \tag{2.8}$$

Un autre type de comparaison que nous présenterons concerne les fonctions de signature. D'après la figure 2.39, à chaque point le long de la courbe, la valeur de la fonction de signature σ correspond à la longueur de l'arc de la courbe de gauche ou, pour une ligne tangente, à ce point. Pour les courbes convexes (entre les points 1 et 6 de la figure 2.39), la fonction de signature prend 1 comme valeur parce qu'à chaque point, toute la courbe suit la tangente gauche de la courbe. L'avantage de cette méthode est le fait qu'elle soit invariante aux translations, rotations et facteurs d'échelle.

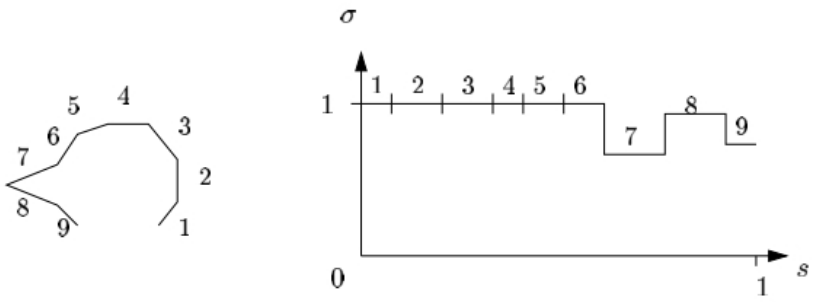


Figure 2.39 – Une courbe et sa fonction de signature d'après [Veltkamp et Hagedoorn, 2001].

Conclusion sur la comparaison des signatures

Cette étape conclut notre démarche de RE. La comparaison des signatures dépend du type de signature. Les travaux qui ont été présentés appartiennent à différents domaines : la théorie des graphes et la géométrie différentielle.

Concernant la comparaison de graphes, il est évident que les approches basées sur les égalités inexactes ne seront pas adaptées à notre problème. En effet, il nous faut un résultat de comparaison qui puisse nous donner la similarité entre deux graphes en comparant chaque nœud de l'un avec chaque nœud de l'autre. Une estimation de cette comparaison, dans le cas de comparaison inexacte, ne nous donnerait pas ce niveau de détails-là. De plus, il serait nécessaire de pouvoir identifier clairement la présence d'un composant (sous-graphe) à l'intérieur d'un autre graphe, d'où le cas d'isomorphisme de sous-graphe. Cependant le temps de comparaison risque d'être important.

Concernant la comparaison géométrique, toutes les solutions présentées dans cette section pourraient convenir à nos travaux notamment dans le cas de reconnaissance de forme dans des images ou dans des mises en plan d'assemblages.

2.4 Conclusion de l'état de l'art

Le but de cette thèse est de proposer une méthode de RE qui combine les résultats de la reconnaissance de forme aux méthodologies KBE en fonction du type de données et de la finalité souhaitée en sortie du processus de RE.

L'état de l'art proposé dans ce chapitre a permis d'exposer les différents travaux existants pour les thématiques suivantes :

— **les techniques de segmentation des données 2D et 3D** : elles permettent de segmenter des images ainsi que des données 3D tels que les nuages de points ou les modèles CAO ;

— **les méthodes de signature** : en fonction du niveau de détails souhaité, plusieurs méthodes répondent aux besoins de notre méthodologie HDI-RE. Le but est de formaliser l'ensemble des caractéristiques permettant de décrire la donnée hétérogène segmentée ;
— **la comparaison des signatures avec une base de connaissances** : la richesse de la base de connaissances peut aider l'utilisateur dans sa démarche de RE surtout lorsque c'est une activité routinière. Un autre avantage de cette base de connaissances sera pour le traitement des données incomplètes, qui pourront tout de même être identifiées, certes partiellement, grâce aux signatures présentes en base.

Parmi toutes les techniques de segmentation proposées, celles basées sur les régions semblent être les plus appropriées à notre problématique car elles correspondent au type d'informations nécessaires pour reconstruire des modèles CAO paramétrés. Pour chaque région (segment), nous pouvons ensuite effectuer différents calculs afin d'extraire des informations topologiques et géométriques. L'idéal serait de réussir à segmenter la donnée en autant de segments que de faces théoriques (nombre de faces du *template* CAO correspondant). Une segmentation par contours ne fournirait, quant à elle, que des informations partielles sur la donnée.

Les méthodes de signature par graphe semblent être celles fournissant le plus haut niveau de détails. Cependant les signatures globales méritent d'être étudiées de part la rapidité d'obtention des résultats. En effet, pour ce type de signature, l'étape de comparaison est négligeable alors que pour les graphes, le temps de calcul risque de poser des problèmes.

Enfin concernant la comparaison des données, l'utilisation d'une base de connaissances contenant certes des signatures mais aussi des *templates* CAO permettra à l'utilisateur de reconstruire la maquette numérique de l'assemblage mécanique. Concernant la comparaison des données, le choix d'une solution de comparaison de graphes dépendra en grande partie, du temps de calcul (sa rapidité en l'occurrence). Les graphes qui seront amenés à être comparés pourraient contenir plusieurs dizaines de nœuds.

Pour conclure, aucune étude existante ne permet de traiter les trois étapes de notre méthodologie ; la grande majorité répondant à seulement un ou deux aspects de notre problématique. La finalité de notre proposition sera donc de :
— combiner une ou plusieurs solutions présentées dans ce chapitre pour chaque étape (segmentation, signature et comparaison) de notre méthodologie ;
— les appliquer à un contexte de RE routinier (capitalisation des connaissances dans une base) ;
— prendre en compte l'existence de données hétérogènes.

La figure 2.40 résume notre proposition ainsi que les éléments de l'état de l'art qui ont été retenus. Notons que seules les solutions concernant les données 3D ont été sélectionnées. En effet, seules les données 3D ont pu être traitées par rapport à la problématique présentée au chapitre 1.

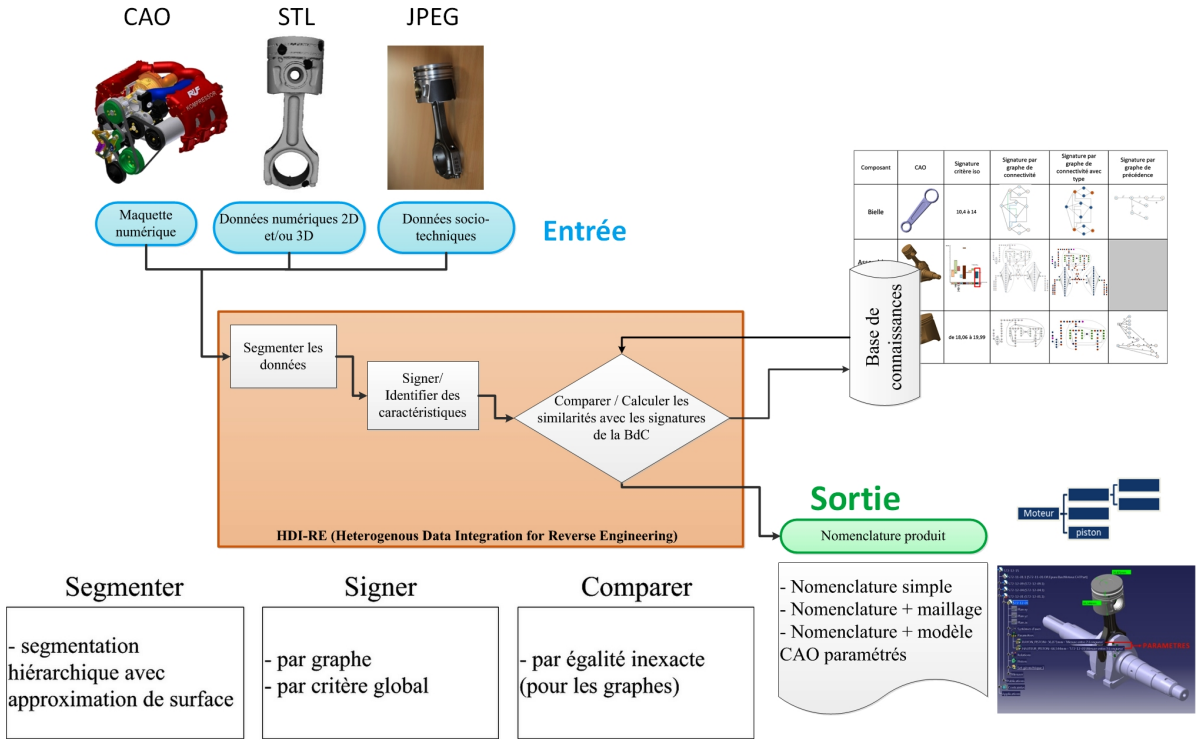


Figure 2.40 – Eléments de l'état de l'art retenus pour notre proposition.

CHAPITRE 3

Proposition : Heterogeneous Data Integration for Reverse Engineering

"On ne peut rien enseigner à autrui. On ne peut que l'aider à le découvrir lui-même."

- Galilée

Introduction

L'état de l'art précédent a permis de présenter les différentes solutions afin de segmenter, de signer des données ainsi que de comparer leur signature. Cependant aucun des travaux présentés ne propose une solution regroupant ces trois étapes et qui intègrerait les notions de données hétérogènes et de données incomplètes. D'où la nécessité de formaliser cela par une méthodologie (HDI-RE) qui sera présentée dans ce chapitre. De plus, l'aspect routinier de l'activité de RE dans notre contexte nécessite de capitaliser les informations extraites au fur et à mesure du processus.

Les travaux des domaines du Shape Matching et du KBE seront adaptés à notre problématique. Ce troisième chapitre se décompose ainsi :

— section 3.1 nous présenterons la méthodologie HDI-RE ;
— section 3.2 nous exposerons la solution de segmentation utilisée pour HDI-RE ;
— section 3.3 nous définirons l'étape de signature et nous révèlerons les différentes solutions proposées ;
— section 3.4 nous présenterons les solutions de comparaison en fonction des types de signature employés.

3.1 La méthodologie HDI-RE

Comme nous l'avons présenté précédemment, la méthodologie HDI-RE (Heterogeneous Data Integration for Reverse Engineering) est composée de trois étapes : la segmentation, la signature des données hétérogènes et la comparaison des signatures avec la base de connaissances.

Au début de ce manuscrit (section 1.2), nous avons identifié trois scénarios industriels d'utilisation du RE :

— "*from scratch*" ;
— "*as designed / as made*" ;
— "*as designed / as maintained*".

Le premier "*from scratch*" a été choisi comme scénario d'étude. En effet, c'est le cas le plus complexe car aucune maquette numérique n'existe, seules les données issues de la numérisation (nuage de points ou photographies) sont à disposition. Ces données ne contiennent aucune information métier comme par exemple l'arborescence dans une maquette numérique. Dans les deux autres scénarios, il s'agit de comparer le produit existant ("*as made/as maintained*") avec sa maquette numérique (la maquette numérique faisant partie des données d'entrée).

Afin de formaliser HDI-RE, nous présentons des diagrammes SADT [Jaulent, 1992] pour le scénario "*from scratch*". Cette méthodologie nous a permis de définir les différentes fonctions Nous énoncerons également les limitations et hypothèses de travail au fur et à mesure du déroulement de la méthodologie :

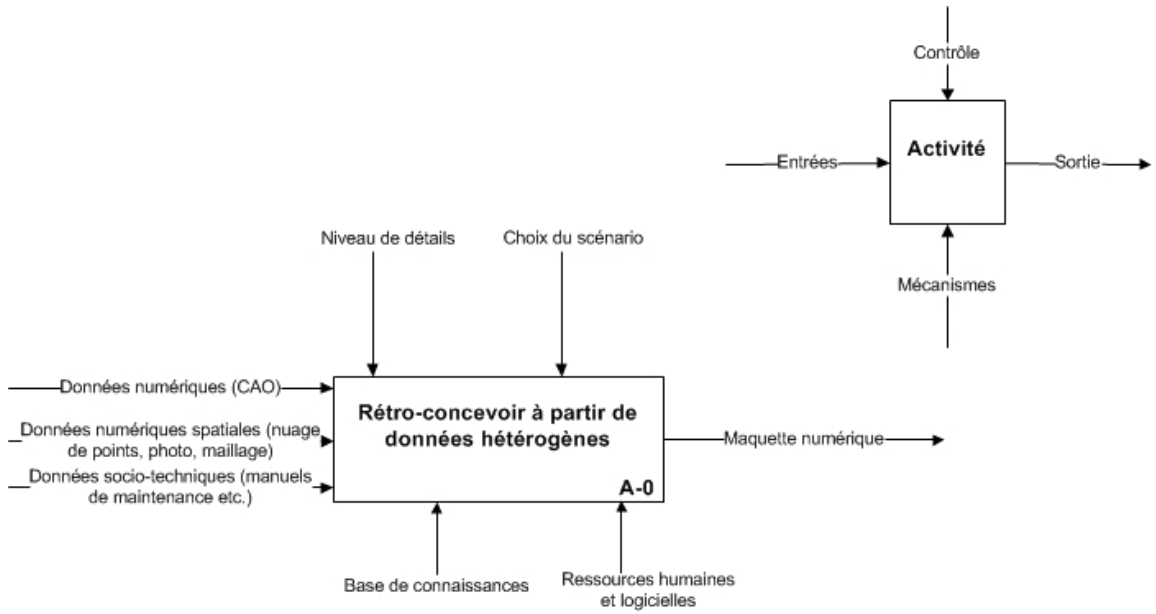


Figure 3.1 – Niveau A0 du diagramme SADT de HDI-RE

L'activité A0 consiste à rétro-concevoir à partir de données hétérogènes en vue de reconstruire une maquette numérique. Les paramètres de cette activité sont :
En entrée :

— les données numériques : il s'agit de la maquette numérique initiale (pour les scénarios "*as designed / as made*" et "*as designed / as maintained*" qui ne sont pas traités dans ce manuscrit) ;
— les données numériques spatiales (données 2D et 3D) tels que des nuages de points, les photographies ou les dessins de définition ;
— les données socio-techniques (données 0D) qui correspondent aux documents textuels tels que les manuels de maintenance.

Limitation 1 Les données socio-techniques ne fournissant qu'un niveau faible d'information, il a été choisi de ne pas les traiter dans cette étude. Les images n'ont pu également être traitées. Les données 3D ont été privilégiées car des informations géométriques et topologiques pouvaient en être rapidement extraites. Les données socio-techniques et 2D font partie des perspectives des travaux et font l'objet de l'étude en cours menée par [Dekhtiar *et al.*, 2016]. Seuls les nuages de points sont utilisés comme donnée d'entrée. On fait l'hypothèse que le nuage de points est déjà maillé au préalable (format .stl binaire ou ascii).

Les contrôles :

- le niveau de détails : il s'agit de celui de la maquette numérique en sortie du processus. Un exemple est donné dans la figure 3.2. Selon le souhait de l'opérateur, il est possible de récupérer soit une nomenclature du produit (image a de la figure 3.2), soit une nomenclature + les enveloppes externes des modèles CAO (image b) soit une nomenclature + des modèles CAO paramétrés (image c). Dans le cas de notre étude, nous considérons le dernier (nomenclature + modèles CAO paramétrés) car c'est le plus complexe.
- le choix du scénario qui correspond aux trois scénarios industriels cités au début de cette section.

Les mécanismes :

- la base de connaissances contient des signatures de différents types des pièces à rétro-concevoir. En effet, on considère qu'un assemblage ne peut être rétro-conçu si la base de connaissances ne contient pas les signatures correspondantes à chacun de ses composants. Il en est de même pour les *templates* CAO qui doivent exister pour que la maquette numérique puisse être reconstruite.
- les ressources humaines et logicielles : nous considérons qu'un seul utilisateur est nécessaire pour se servir de la méthodologie de RE proposée dans ce manuscrit.

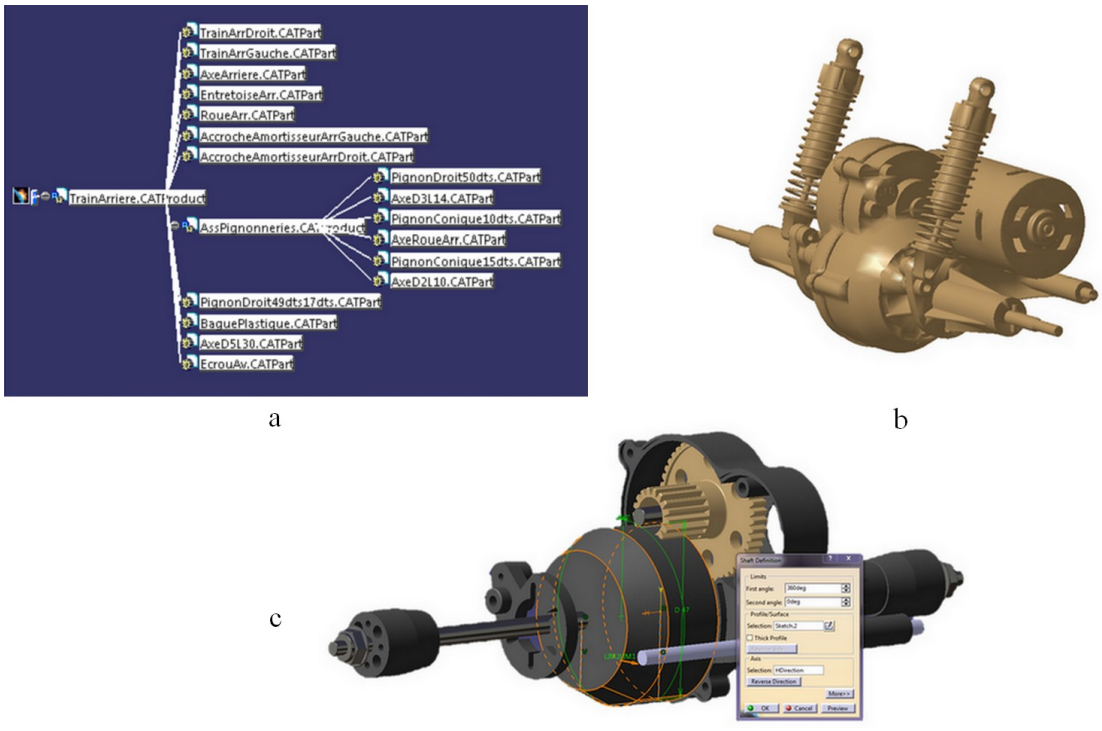


Figure 3.2 – Différents niveaux de détails pour une maquette numérique dans CATIA V5-6R2013 : (a) avec une nomenclature de l'assemblage, (b) avec une nomenclature + les enveloppes externes des pièces, (c) une nomenclature + les modèles CAO paramétrés.

Hypothèse 1 On considère l'existence de dessins de définition dans la base de connaissances sous forme numérique ainsi que le graphe de précédence qui leur est associé. C'est une représentation formelle

des intentions de conception utilisée en bureau des méthodes. [Ali, 2015] le définit comme un "graphe de liaisons ou d'antériorité de surfaces où les exigences dimensionnelles et géométriques sont représentées. Ces exigences sont issues des connaissances de l'utilisateur et de règles de fabrication". Un exemple est donné dans la figure 3.3 avec la mise en plan d'une bielle (a), l'identification de ses surfaces (b) et son graphe de précédence correspondant (c). Dans le cadre de nos travaux, le graphe de précédence est sous la forme d'un fichier .xml. Ce type de graphe sera utile dans l'étape de signature et comparaison des signatures (sous-activités de l'activité A2). Ce graphe pourrait faire partie des données d'entrée cependant son importation doit se faire en amont de l'activité de RE (phase de capitalisation).

La richesse de la base de connaissances est importante. Tout type de signature confondu, nous supposerons qu'il n'est pas possible d'identifier un composant s'il n'existe aucune signature de même type correspondant à ce composant dans la base.

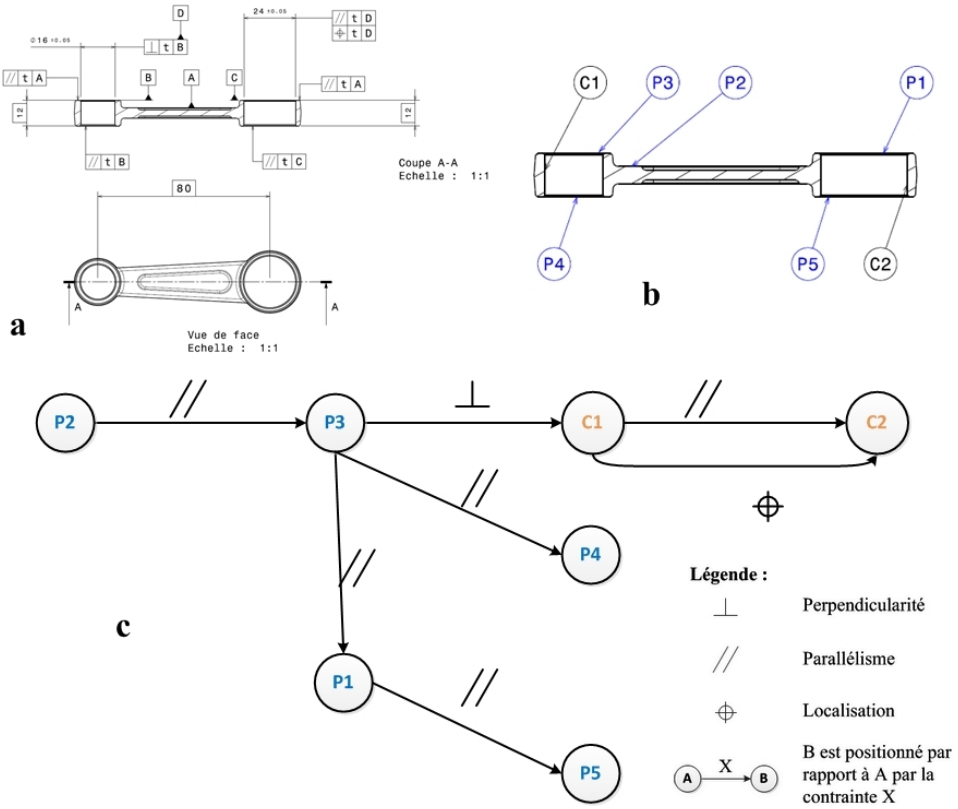


Figure 3.3 – Dessin de définition d'une bielle et son graphe de précédence.

Limitation 2 Le logiciel Teexma^{TM} est utilisé afin d'organiser les signatures dans la base de connaissances.
Les sorties :

— il s'agit de la maquette numérique de l'assemblage mécanique qui peut être sous différentes formes selon le niveau de détails : une simple nomenclature ou une nomenclature + les enveloppes extérieures des pièces ou une nomenclature + les modèles CAO paramétrés des composants de l'assemblage.

L'activité A0 se décompose ensuite en 4 sous-activités explicitées dans la figure 3.4.

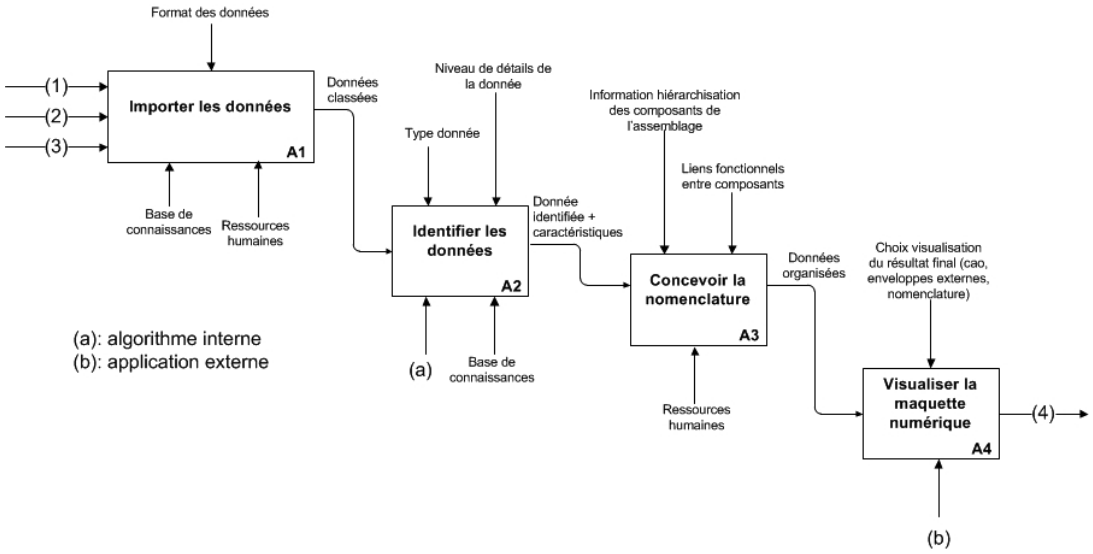


Figure 3.4 – Décomposition de l'activité A0 du processus global HDI-RE

A1 : Importer les données

Cette première activité consiste à importer des données dans notre système et à les classer selon leur type.
Les entrées :

— (1), (2), (3) correspondent aux 3 entrées décrites dans l'étape A0 (figure 3.1).

Les contrôles :

— le format des données doit être connu par le système. Par exemple, une donnée de type image ne peut pas être identifiée (étape A2) si aucun mécanisme de signature d'image n'existe.

Les ressources :

— les ressources humaines restent identique à A0 : l'utilisateur sélectionne les données à importer dans le système.
— la base de connaissances décrite précédemment.

La sortie :

— les données classées : les données sont classées selon leur type (nuages de points, images, etc.)

Hypothèse 2 L'activité A1 n'est pas traitée dans ces travaux. Sa mise en œuvre est réalisée par la société DeltaCAD, le porteur du projet.

A2 : Identifier les données

Cette activité correspond au cœur des travaux présentés dans ce manuscrit. Il s'agit d'identifier chaque donnée en entrée du système et d'en extraire un ensemble de caractéristiques.
L'entrée :

— pour chaque donnée importée, l'activité A2 va permettre de l'identifier.

Les contrôles :

— le type de donnée (2D ou 3D) va déterminer le mécanisme d'identification à utiliser.
— le niveau de détails de la donnée influe sur le mécanisme d'identification. Par exemple, un nuage de points peut apporter des informations topologiques et géométriques, nous pouvons alors utiliser un mécanisme d'identification employant ce type d'information.

Les ressources :

— la base de connaissances permet d'identifier le composant. La signature de la donnée à identifier est comparée à toutes les signatures de la base de connaissances.
— un ou plusieurs algorithmes internes permettent d'identifier la donnée.

La sortie :

— la donnée est identifiée en totalité ou partiellement. Selon la taille de la donnée et sa complétude, seulement une partie de la donnée est identifiée. Cela dépend également des signatures présentes dans la base de connaissances. Pour les données identifiées, on récupère également les caractéristiques géométriques et topologiques de chaque composant.

A3 : Concevoir la nomenclature de l'assemblage

L'activité A3 correspond à l'étape clé du processus de rétro-conception d'un assemblage mécanique. Après avoir identifié différents composants dans les données (activité A2), il faut désormais reconstruire la structure de l'assemblage.
L'entrée :
— chaque donnée est étiquetée avec l'identification de un ou plusieurs composants dont on possède les caractéristiques.
Les contrôles :
— les composants sont hiérarchisés dans l'arbre par rapport à leur fonction et aux autres composants qui les entourent.
— les liens fonctionnels correspondent aux fonctions mécaniques. Par exemple, pour assurer un pivot, un arbre est en contact avec une pièce possédant un alésage. Les caractéristiques de ces deux pièces sont alors liées.

Les ressources :
— c'est l'utilisateur qui construit cette nomenclature dans une interface dédiée (solution développée par DeltaCAD également).
La sortie :
— les données appartenant au même composant sont regroupées dans la nomenclature mais aussi dans l'espace (pour la visualisation).

A4 : Visualiser la maquette numérique

L'activité A4 permet de concrétiser la démarche de RE en affichant la maquette numérique de l'assemblage rétro-conçu.
L'entrée :
— les données sont organisées et les caractéristiques extraites sont utilisées pour la visualisation de la maquette numérique (instanciation des *templates* CAO).
Les contrôles :
— la visualisation dépend du choix de l'utilisateur. Comme expliqué dans notre cas, on choisit d'étudier le cas où l'utilisateur souhaite récupérer une maquette numérique composée de modèles CAO paramétrés.
Les ressources :
— la visualisation est faite dans une application externe à notre système (modeleur CAO par exemple).
La sortie :
— cette sortie est identique à celle de l'activité A0.

Hypothèse 3 Les activités A3 et A4 ne seront pas traitées et elles seront également développées par DeltaCAD.

Seule l'activité A2 (Identifier les données) est détaillée ci-après car c'est sur celle-ci que se concentrent nos travaux. Les trois sous-activités qui en découlent sont présentées dans la figure 3.5.

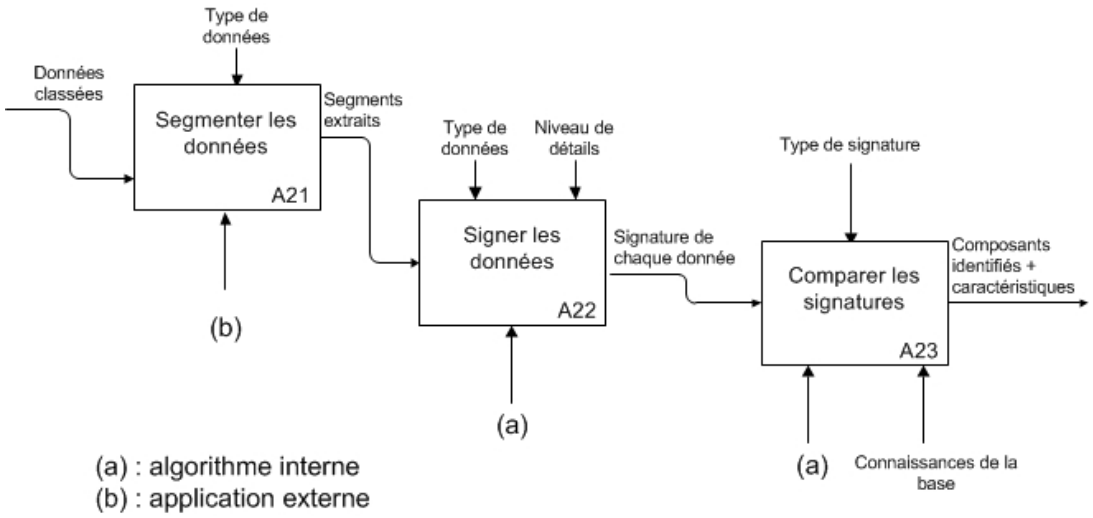


Figure 3.5 – Décomposition de l'étape A2 du processus global HDI-RE.

A21 : Segmenter les données

L'activité A21 permet de segmenter des données d'entrées, afin d'extraire des informations de chaque segment.
L'entrée :

— il s'agit des données d'entrée. On précise qu'elles ont été identifiées dans le sens où elles ont un identifiant unique comme dans les bases de données.

Les contrôles :

— le type de données influe sur l'algorithme de segmentation utilisé.

Les ressources :

— une application externe est appelée (en mode batch) pour segmenter les données. De nombreuses solutions existent sur le marché et répondent à notre besoin en segmentation.

La sortie :

— chaque segment extrait est ensuite analysé. Le traitement des données se trouve alors simplifié.

A22 : Signer les données

L'activité A22 consiste à signer chaque donnée. Chaque signature est un ensemble de caractéristiques (ex : agencement des segments pour créer un motif, calcul d'aire ou de volume sur un ou plusieurs segments) permettant de décrire la donnée.
L'entrée :

— chaque segment extrait de chaque donnée est signé grâce à un mécanisme de signature qui lui est propre.

Les contrôles :

— le type de données joue un rôle sur le mécanisme de signature. Pour chaque type de données, un mécanisme unique est utilisé.

— selon le résultat souhaité en sortie du processus tel que décrit en A0, le mécanisme de signature dépend du niveau de détails désiré. Par exemple, si la sortie souhaitée est une nomenclature, il n'est pas nécessaire d'utiliser un mécanisme de signature faisant appel aux caractéristiques géométriques présentes dans les données d'entrée. Une signature globale (caractéristique globale) peut alors suffire.

Les ressources :

— chaque algorithme de signature est développé en interne. En fonction du type de donnée et du niveau de détails souhaité, un plugin différent est lancé lors de chaque signature

La sortie :

— la signature de chaque donnée correspond à un ensemble de caractéristiques qui permettent de décrire la donnée.

A23 : Comparer les signatures

L'activité A23 consiste à comparer les signatures issues de A22 avec celles de la base de connaissances. Le score de similarité entre deux signatures permet d'identifier une donnée.

L'entrée :

- toutes signatures de chaque donnée sont comparées aux signatures de même type de la base de connaissances.

Les contrôles :

- le type de signature définit le mécanisme de comparaison : la comparaison de signature par graphe n'est pas la même pour les signatures globales par exemple.

Les ressources :

- chaque signature possède un algorithme de comparaison qui lui est propre. Il fonctionne en interne.
- la base de connaissances est un pilier de cette étape. Les résultats de comparaison ne peuvent être satisfaisants si la base de connaissances n'est pas suffisamment riche.

La sortie :

- un composant est identifié dès lors qu'une des signatures des données d'entrée est en correspondance avec une signature de la base de connaissances. Plusieurs composants peuvent être identifiés dans une même donnée puisqu'il s'agit de RE d'assemblages mécaniques.

Conclusion sur la méthodologie

Cette section a permis de dérouler tout le processus de la méthodologie de RE dont il est question dans ces travaux et de situer HDI-RE par rapport à celui-ci. Ainsi nos travaux porteront essentiellement sur les étapes A21, A22 et A23 et en particulier l'étape de A22 qui constitue le cœur de cette étude.

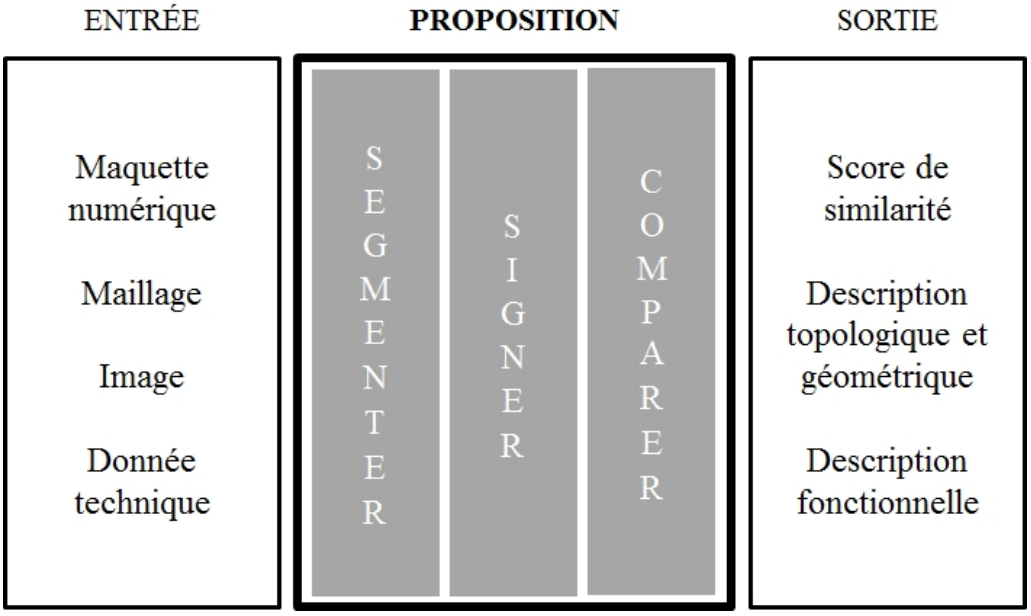


Figure 3.6 – Synthèse de notre proposition par rapport au contexte des travaux.

La figure 3.6 illustre notre proposition HDI-RE par rapport aux entrées et sorties détaillées dans les diagrammes présentés précédemment. Pour chaque entrée (colonne de gauche), il peut y avoir une ou plusieurs sorties possibles (colonne de droite). Par exemple, pour obtenir une maquette numérique avec des modèles CAO paramétrés, il nous faudra obtenir une description topologique et géométrique de la pièce ainsi qu'une description fonctionnelle. Le chemin pour aller de la gauche vers la droite (c'est-à-dire la proposition), ce sont les différentes solutions présentées dans les sections suivantes.

Un autre exemple concerne la comparaison de nomenclatures de maquettes numériques, la sortie sera alors sous la forme d'un score de similarité. Une solution par signature globale sera alors envisagée.

Les sections suivantes traiteront ainsi des étapes de segmentation (section 3.2), de signature (section 3.3) et de comparaison des signatures avec la base de connaissances (section 3.4). Un scénario complet sera présenté dans le dernier chapitre, où un cas d'étude sera illustré de la segmentation des données jusqu'à l'instanciation des modèles CAO paramétrés pour la maquette numérique.

3.2 La segmentation des données

Cette section concerne l'activité A21 présentée dans la section précédente. Comme expliqué en amont, il s'agit de segmenter uniquement des données de type nuage de points et plus exactement les maillages.

La sélection de la solution de segmentation a été effectuée suite à l'étude de deux logiciels libres : CGAL [CGAL Project, 2015] et Efpisoft ([Attene *et al.*, 2006]).

Le logiciel choisi est Efpisoft. Il repose sur une méthode de segmentation par région et particulièrement par clusterisation hiérarchique. Il permet de traiter les nuages de points préalablement maillés (dans le format ASCII). Le principe de ce logiciel est le suivant : chaque triangle du maillage est considéré comme un cluster. Un graphe est alors établi avec pour nœud chaque cluster. Une arête est créée entre chaque nœud si les triangles en question partagent une arête. Ensuite, l'opération de segmentation par région consiste à fusionner successivement chaque triangle à un autre triangle ou groupe de triangles existant (région). Pour cela, des arêtes du graphe sont "contractées". L'ordre dans lequel sont contractées les arêtes dépend de ce qu'on appelle le "coût" de contraction. Une file d'attente est créée et remise à jour après chaque fusion. C'est l'arête avec le coût le plus faible (en haut de la file) qui est contractée. La figure 3.7 illustre cette méthode. Dans la figure (a), on aperçoit le maillage et son graphe correspondant.

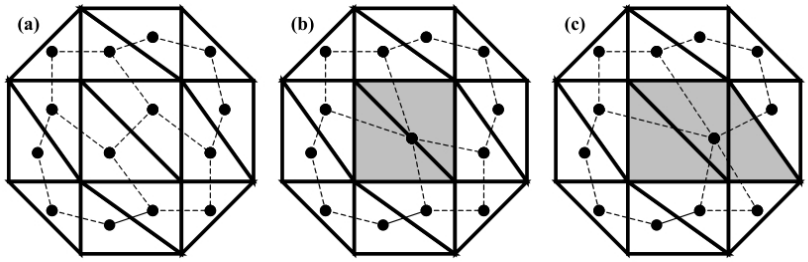


Figure 3.7 – Contraction de triangles par hiérarchie d'après [Attene *et al.*, 2006].

Ensuite, une première arête est contractée en (b). La surface grise représente le nouveau cluster et ainsi de suite pour (c) où un nouveau triangle est ajouté au cluster précédent. Le processus est arrêté lorsqu'il n'y a plus qu'un seul cluster (c'est-à-dire un seul segment). La particularité des travaux de [Attene *et al.*, 2006] réside dans le calcul du coût de contraction. Chaque cluster est ajusté à un modèle (plan, sphère ou cylindre). L'algorithme calcule pour chaque contraction possible :

1. les coefficients du plan le mieux ajusté
2. les coefficients de la sphère la mieux ajustée
3. les coefficient du cylindre le mieux ajusté.

Chaque erreur relative (considérée comme coût de contraction) est approximée et la valeur minimale est gardée et placée en haut de la file d'attente. Ainsi plus le nombre de clusters est grand, plus le nombre de clusters de type plans augmente jusqu'à que ce nombre soit égal au nombre de triangles.

Efpisoft compile toutes les configurations possibles (variation du nombre de clusters) et il est possible de les visualiser grâce à une barre de défilement.

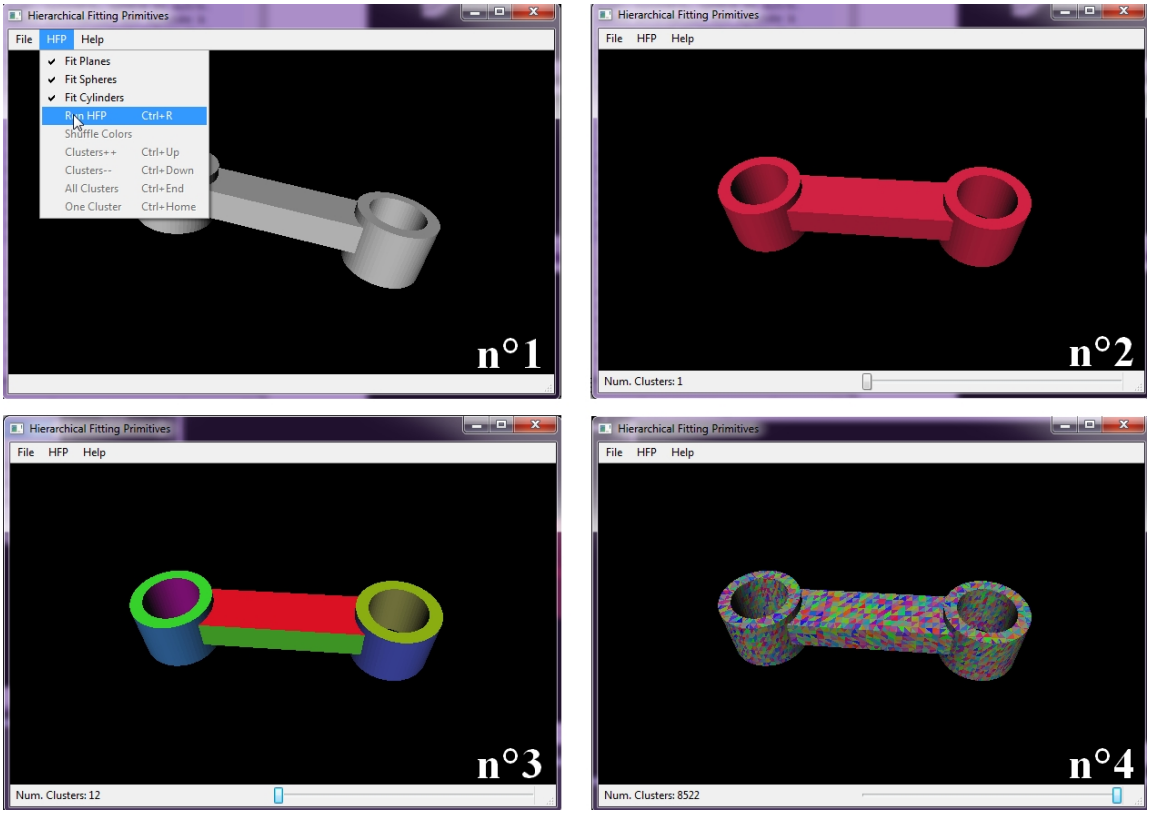


Figure 3.8 – Segmentation sous Efpisoft et affichage du résultat.

La segmentation est lancée par la commande "Run HPF" (image n°1 de la figure 3.8) et le résultat s'affiche ensuite avec 1 cluster (image n°2). Le nombre peut être augmenté par l'utilisateur jusqu'à ce que le nombre de clusters corresponde au nombre de faces (12 clusters sur l'image n°3). Puis nous retrouvons la dernière configuration possible sur l'image 4, où le nombre de clusters est égal au nombre de triangles (8522 pour cette bielle).

Nous remarquerons que la segmentation dans la figure n°3 est parfaite. En effet, nous avons choisi de montrer un exemple simple en utilisant une donnée issue de la CAO (fichier converti en .stl depuis le logiciel de conception). Dans le cas de pièces plus complexes (avec des arrondis, des chanfreins etc.), il est difficile d'obtenir un nombre de clusters égal au nombre de faces. Il en va de même pour les données scannées qui présentent des irrégularités ou encore des discontinuités dans le maillage. Une étude approfondie a été menée sur l'influence du nombre de clusters par rapport au processus de signature. Elle sera présentée ultérieurement.

Une fois le nombre de clusters choisi (12 dans notre cas), la segmentation est enregistrée sous la forme d'une scène de format Open Inventor (.iv). Afin de gérer l'interopérabilité entre les différents logiciels (segmentation et signature), un plugin a été créé afin de convertir ce fichier en format .xml. Le fichier XML correspondant à la bielle présentée dans la figure 3.8 est disponible en annexes de ce manuscrit (voir annexe A.1).

3.3 La signature des données

Dans la littérature, les mécanismes de signature sont utilisés pour la reconnaissance de forme. De nombreux travaux existent incluant le domaine de la mécanique et en particulier pour la recherche de modèles CAO dans des bases de données. Nous nous sommes appuyés des travaux de [Tangelder et Veltkamp, 2007] présentés dans le chapitre précédent. Grâce à leur classification des méthodes, nous avons étudié deux d'entres elles : l'une à partir d'un critère global et l'autre avec un graphe ; celles par géométrie étant principalement dédiées aux données 2D (images). Nous présenterons dans un premier temps celle basée sur les graphes car elle s'insère complètement dans notre processus de RE. En effet, la donnée doit d'abord être segmentée avant d'être signée. Puis le graphe (signature) doit être comparé avec tous les autres graphes de la base de connaissances afin d'identifier les composants présents dans la donnée d'entrée. Dans un deuxième temps, nous présenterons la signature de type global. Pour ce cas-là, la donnée n'a pas besoin d'être segmentée au préalable et la comparaison ne requiert aucune ressource spécifique (comparaison de valeurs).

Avant de présenter chaque type de signature, il convient de formaliser une définition du terme "signature" : **c'est un ensemble de caractéristiques permettant de décrire un objet. Comme un objet peut être représenté par plusieurs médias (sources), il peut avoir différentes signatures. De plus, pour une et même source (donnée d'entrée), plusieurs mécanismes de signature peuvent être utilisés séparément ou combinés.**

3.3.1 Les signatures par graphe

Dans cette section, nous présentons un mécanisme de signature dédié aux données de type nuage de points. Cette étape arrive dans la continuité de celle de segmentation. Il s'agit donc de signer un nuage de points maillé à l'aide d'un graphe.

Notre méthode repose sur une signature à trois niveaux dont chacun d'entre eux correspond à un niveau de détails. Plus le niveau est élevé, plus la quantité d'informations fournie par le graphe est importante. De plus, les niveaux 2 et 3 dépendent du niveau qui les précède. La figure 3.9 illustre le processus de signature. Nous allons tout d'abord présenter chaque niveau :

1. premier niveau de signature : chaque nœud représente un cluster (aussi appelé segment) du maillage. Les arêtes réfèrent aux liens de connectivité entre les clusters. Chaque lien correspond à une arête commune (contact) entre deux clusters. Considérons G_1 ce premier niveau de graphe avec N_1 l'ensemble des nœuds et E_1 ses arêtes connectées. D'après la figure 3.10 représentant le graphe de premier niveau d'une bielle, nous avons : $G_1 = \{N_1, E_1, \Psi\}$ avec $N_1 = \{1, 2, 3, 4, 5, ...12\}$ et $E_1 = \{e1, e2, e3, e4, ...e_n\}$ et Ψ l'ensemble des liens de connectivité entre l'ensemble des nœuds et l'ensemble des arêtes.

2. deuxième niveau de signature : le graphe de premier niveau est réutilisé. Chaque cluster (nœud du graphe) est classé selon son type : plan, cylindre, sphère ou autre. Cette information est ajoutée à chaque nœud en tant qu'attribut. Cette deuxième signature correspond au graphe G_2 avec N_2 ses nœuds, E_2 ses arêtes et $A = \{A_1, A_2, A_3, A_4\}$ les attributs possibles pour chaque nœud (A_1 pour plan, A_2 pour cylindre, A_3 pour sphère et A_4 pour autre). On a alors : $G_2 = \{N_2, E_2, \Psi\}$, Ψ restant identique au premier niveau. Et d'après les travaux de [Jusufi, 2013], les attributs des nœuds sont représentés par A_i une colonne dans la table des attributs $A = (a_{ij})(j = 1...|N_2|; i = 1...4)$ et contient une valeur attribuée par nœud. La figure 3.10 illustre également le graphe de niveau 2. Une lettre a été ajoutée devant chaque numéro de nœud : P pour un cluster de type plan, C pour cylindre et S pour sphère et cela afin de représenter les attributs de chaque nœud.

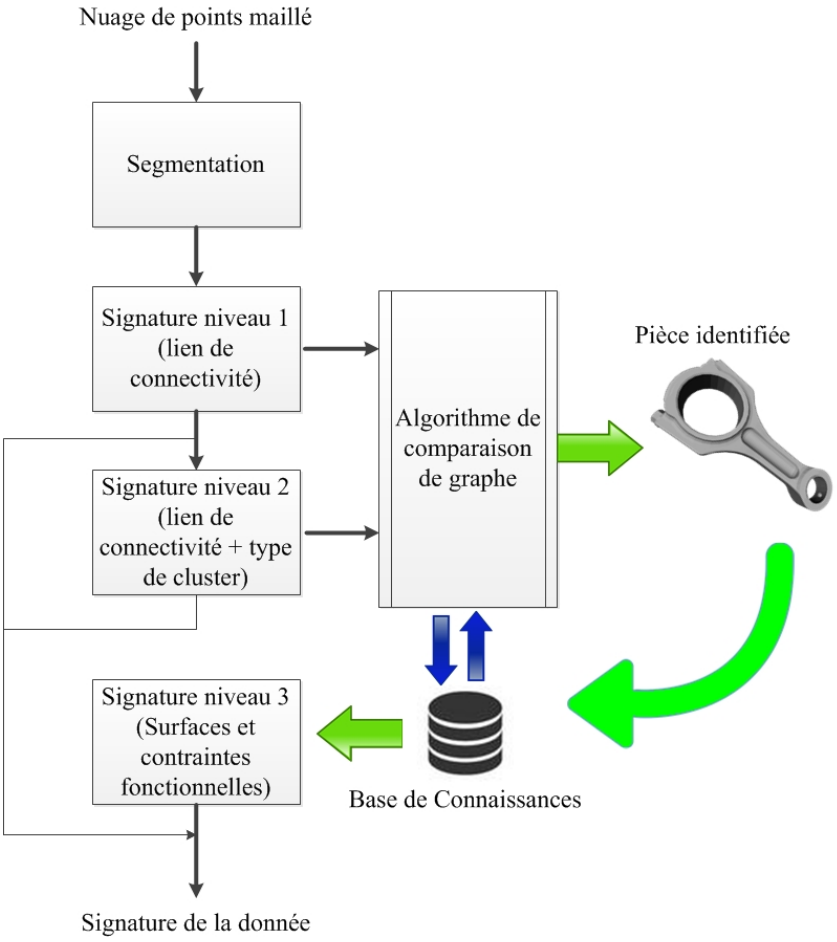


FIGURE 3.9 – Signature à trois niveaux par graphe.

3. troisième niveau de signature : celui-ci est dépendant des niveaux qui le précèdent mais aussi des résultats de la comparaison de niveau 2 (expliqué plus tard). Au lieu d'avoir uniquement une donnée hétérogène en entrée du processus, une ou plusieurs signatures de la base de connaissances sont utilisées pour signer. Ce graphe de niveau 3 est construit en comparant la signature de niveau 2 (obtenue précédemment) et le graphe de niveau 3 du ou des composants qui ont été identifiés suite à la comparaison de niveau 2. La construction de G_3 sera plus amplement détaillée dans la section 3.4. Nous considérons G_3 ce graphe avec N_3 et E_3 ses nœuds et arrêtes ainsi que Ψ' qui est l'ensemble des liens de connectivité entre l'ensemble de nœuds N_3 et E_3 l'ensemble d'arêtes du graphe. Les liens de connectivité de ce graphe sont différents des graphes G_1 et G_2 ($\Psi \neq \Psi'$).

   Ce graphe est appelé graphe de précédence et il est orienté [1]. Un exemple a été donné dans la figure 3.3. Ce niveau de graphe est important car il permet l'identification des surfaces fonctionnelles de la donnée d'entrée. Les informations extraites de ces surfaces seront utiles, dans une étape en aval de notre processus, pour reconstruire une maquette numérique avec des modèles CAO paramétrés ; les paramètres étant valués grâce aux caractéristiques géométriques extraites (diamètre d'un cylindre par exemple).

   Les arêtes de ce graphe de troisième niveau ont la particularité d'avoir des labels qui correspondent aux contraintes géométriques et il en existe différents types. Dans nos travaux, seules les contraintes de perpendicularité, de coaxialité, de localisation, d'inclinaison et de parallélisme seront employées (d'après la norme ISO GPS). Les nœuds, quant à eux, gardent les mêmes attributs qu'au deuxième niveau. Enfin, le nombre de nœuds de ce graphe diffère cependant par rapport au niveau 2.

1. Graphe orienté : les arêtes de ce graphe sont des flèches orientées.

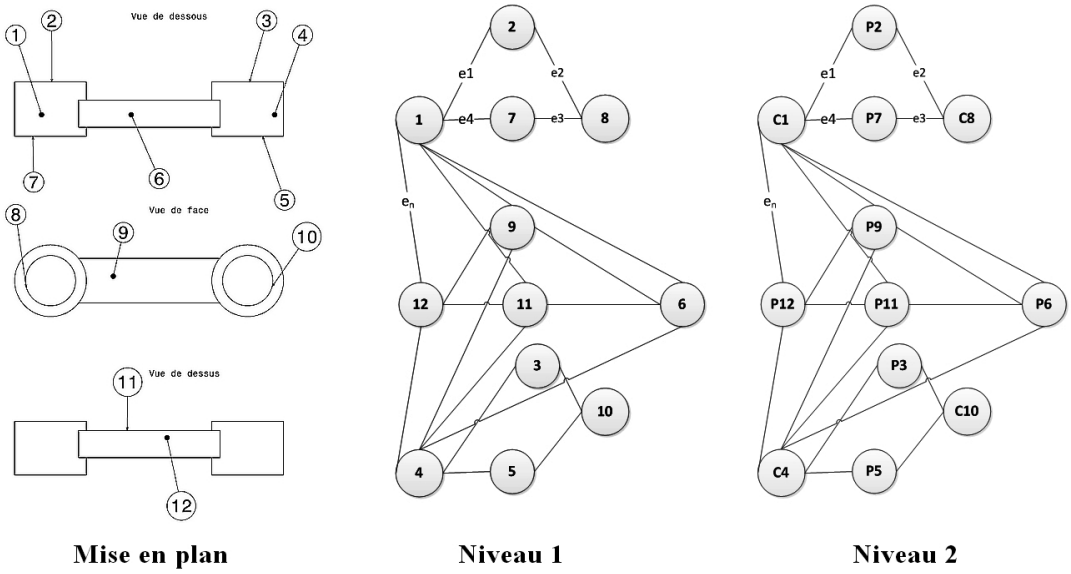


Figure 3.10 – Signature par graphe de niveau 1 et 2 d'une bielle. La donnée utilisée a été simplifiée pour plus de clarté.

En effet, le nombre de nœuds dans un graphe de précédence est inférieur au nombre de nœuds dans un graphe de niveau 2 (seuls les nœuds correspondant aux surfaces fonctionnelles existent). L'avantage d'utiliser ce graphe repose dans le faible nombre de nœuds. En effet, nous aurions pu reprendre le graphe de niveau 2 et lui ajouter toutes les contraintes fonctionnelles entre tous ses nœuds, cependant le résultat aurait était un graphe très "chargé". Ainsi ce graphe de niveau 3 propose uniquement le juste nécessaire : c'est-à-dire les nœuds relatifs aux surfaces fonctionnelles ainsi que les contraintes qui en découlent.

Les étapes de comparaison seront expliquées à la section 3.4 du manuscrit.

Après avoir expliqué le principe de cette signature, nous allons maintenant présenter son "fonctionnement" et nous exposerons cinq scénarios qui découlent des trois niveaux de signature présentés dans la figure 3.9. Ces scénarios présentent les différentes manières dont l'utilisateur peut signer les données d'entrée, en fonction du type d'information qu'il souhaite récupérer en sortie (informations topologiques, géométriques ou fonctionnelles). Pour les décrire, la figure 3.9 a été déclinée en plusieurs versions dont la description est faite ci-après.

Scénario n°1 : signature et comparaison de niveau 1. L'utilisateur souhaite signer et comparer avec le premier niveau (figure 3.11). Les informations extraites sont uniquement de type "connectivité". Suite à la comparaison de niveau 1 (étape n°3) avec la base de connaissances, une ou plusieurs pièces sont identifiées. Le scénario n°1 s'arrête après l'affichage des résultats de la comparaison (score de similarité entre les signatures).

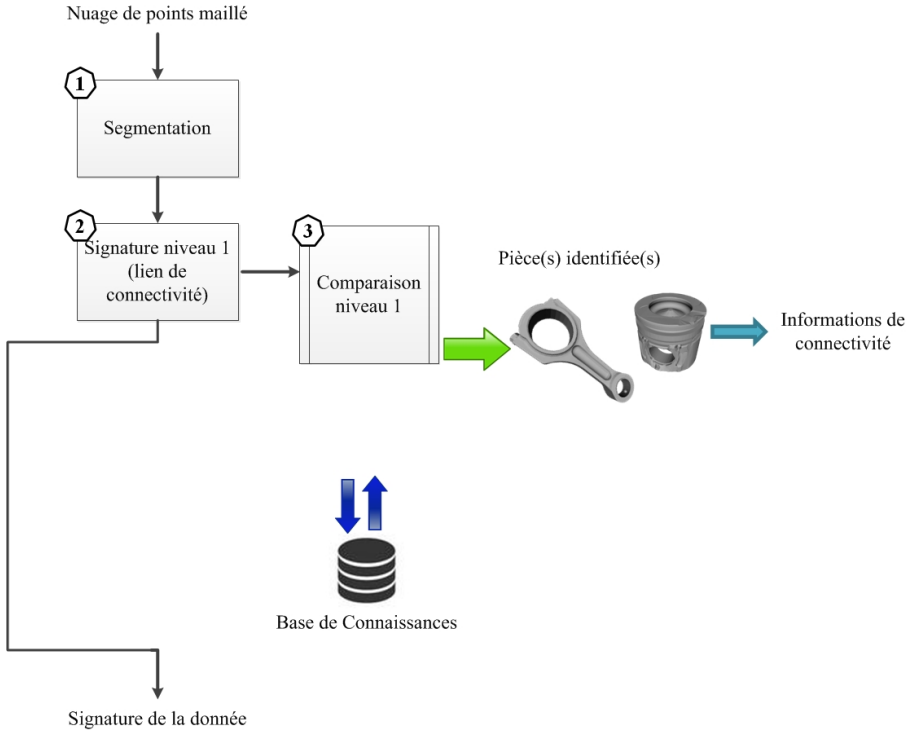


Figure 3.11 – Scénario n°1 : signature et comparaison de niveau 1.

Scénario n°2 : signature et comparaison de niveaux 1 puis 2. Ce scénario est la suite du premier. Notons que la comparaison de niveau 1 (étape n°3 de la figure 3.12) est réalisée avec toutes les signatures de la base de connaissances. Une fois que l'utilisateur a généré la signature de niveau 2 (étape n°4), la signature obtenue n'est pas comparée à toutes les signatures de même niveau de la base mais uniquement aux signatures "sélectionnées" suite à la comparaison de niveau 1. Cette sélection est manuelle. L'avantage est de réduire le volume de signatures à comparer pour le niveau 2 et ainsi réduire le temps de calcul. L'utilisateur choisira ainsi les meilleures signatures (c'est-à-dire le ou les composants correspondants) selon deux critères : le score de similarité et le nombre de clusters (nombre de nœuds) de la signature. C'est ensuite la (les) signature(s) de niveau 2 correspondante(s) au(x) composant(s) sélectionné(s) qui est (sont) utilisée(s) dans l'étape n°5. De plus amples informations seront apportées lors de la mise en situation avec un cas industriel dans la section 4.4. En sortie de ce scénario, l'utilisateur possède alors deux signatures de ses données d'entrée (niveau 1 et 2) ainsi qu'une liste des composants avec leur score de similarité obtenu. Nous pourrons ainsi considérer qu'un composant est "identifié" dès lors que la valeur de son score est plus élevée que les autres.

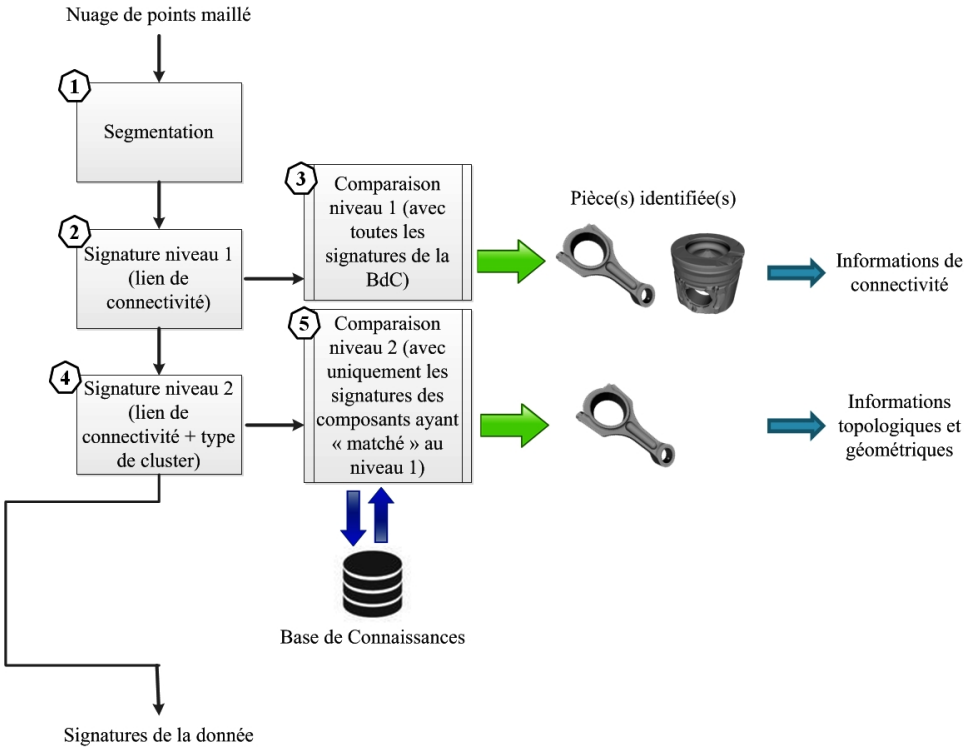


FIGURE 3.12 – Scénario n°2 : signature et comparaison de niveau 1 puis 2.

Scénario n°3 : signature et comparaison de niveaux 1 puis 2 puis 3. Ce scénario est la suite chronologique du précédent, c'est le cas où l'utilisateur souhaite poursuivre le processus de RE en vue de récupérer des informations fonctionnelles et ainsi reconstruire une maquette numérique avec des modèles CAO paramétrés. Suite à la comparaison de niveau 2 (étape n°5 de la figure 3.13), l'utilisateur sélectionne un des composants (étape n°6) de la même manière qu'expliqué précédemment. Cependant son choix peut être restreint du fait du peu de signatures de la base utilisées pour cette comparaison.

A l'étape n°7, il s'agit de récupérer dans la base la signature de niveau 3 correspondante au composant identifié (sélectionné). Puis notre donnée d'entrée est signée avec le niveau 3 (cette étape de signature est particulière. Elle est détaillée à la section 3.4.3). Puis les étapes 7 et 8 sont reproduites autant de fois que l'utilisateur sélectionne de composants à l'étape n°6. En sortie de ce scénario, l'utilisateur possède les signatures de niveaux 1 et 2 de la donnée d'entrée ainsi que la signature de niveau 3.

Ce scénario est le plus complet de tous et c'est celui qui sera utilisé pour l'application à un cas industriel dans la section 4.4.

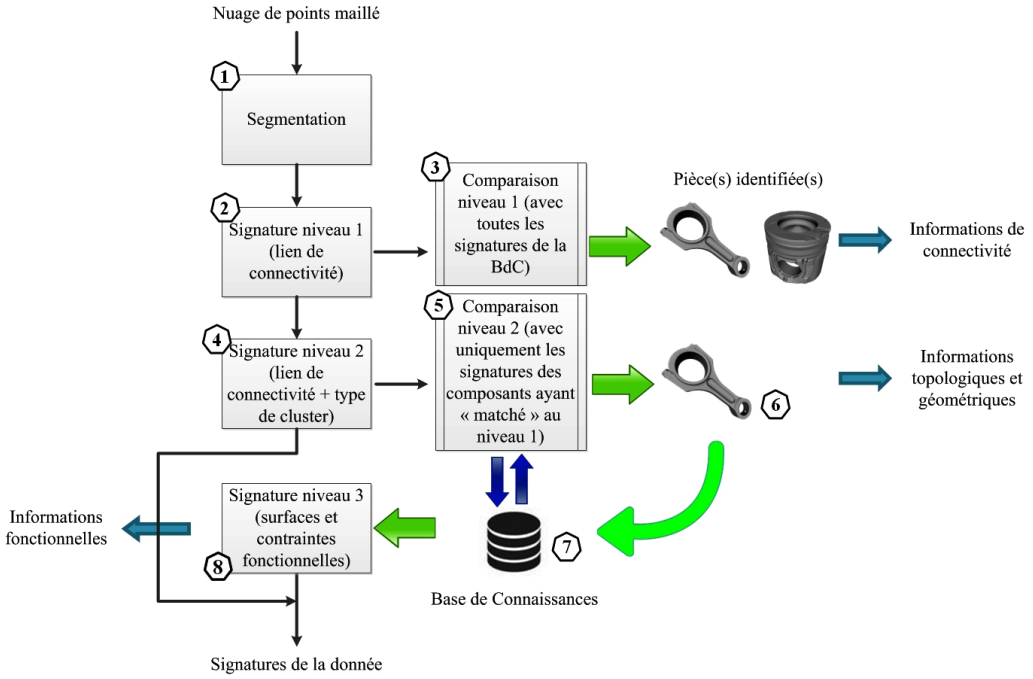


FIGURE 3.13 – Scénario n°3 : signature et comparaison de niveaux 1 puis 2 puis 3.

Scénario n°4 : signature de niveaux 1 et 2 puis comparaison de niveau 2. Pour ce scénario, l'utilisateur signe successivement ses données d'entrée avec la signature de niveau 1 puis avec le niveau 2 (étapes n°2 et 3 de la figure 3.14). Notons qu'il est impossible dans notre cas, de signer directement avec le niveau 2 car cette signature s'appuie sur celle de niveau précédent.

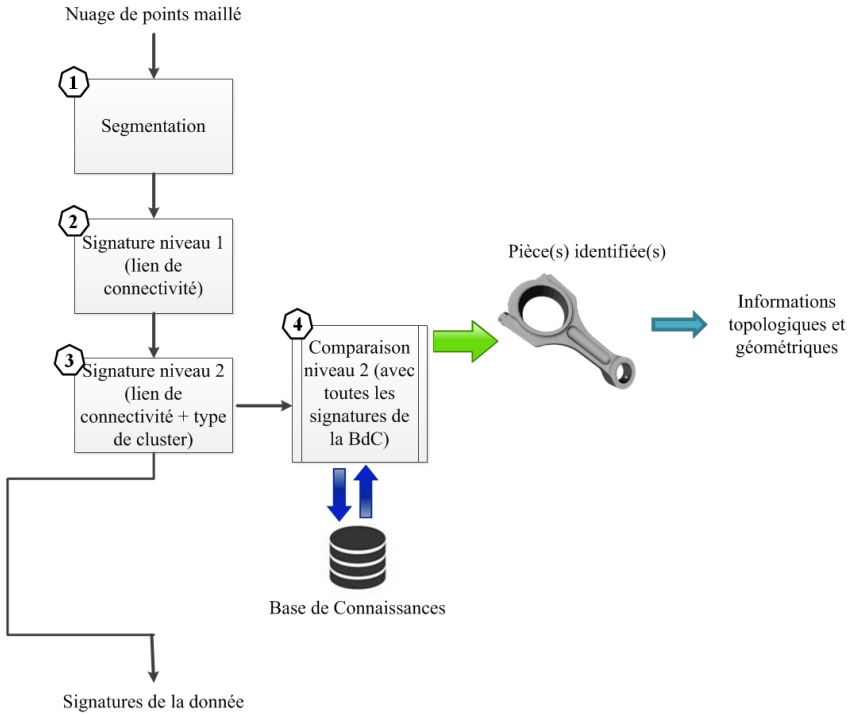


FIGURE 3.14 – Scénario n°4 : signature de niveaux 1 et 2 puis comparaison de niveau 2.

La signature de niveau 2 de la donnée d'entrée est comparée (étape n°4) à l'ensemble des signatures de même niveau de la base de connaissances. Le scénario s'arrête dès l'affichage des scores de similarité entre les signatures comparées. En sortie, l'utilisateur dispose d'une liste de composants avec leur score de similarité respectif ainsi que des signatures de niveaux 1 et 2 de la donnée d'entrée (nuage de points maillé).

Scénario n°5 : signature de niveaux 1 et 2 puis comparaison de niveau 2 puis signature de niveau 3. Ce scénario est similaire au n°3 pour ce qui concerne les trois dernières étapes (étapes n°5, 6 et 7 de la figure 3.15). Ce scénario est également utilisé dans le cas où l'utilisateur souhaite récupérer des informations fonctionnelles. Il dispose en sortie du processus d'une liste de composants avec leur score de similarité respectif pour la comparaison de niveau 2 (étape n°4) ainsi que des signatures de la donnée d'entrée pour les niveaux 1, 2 et 3.

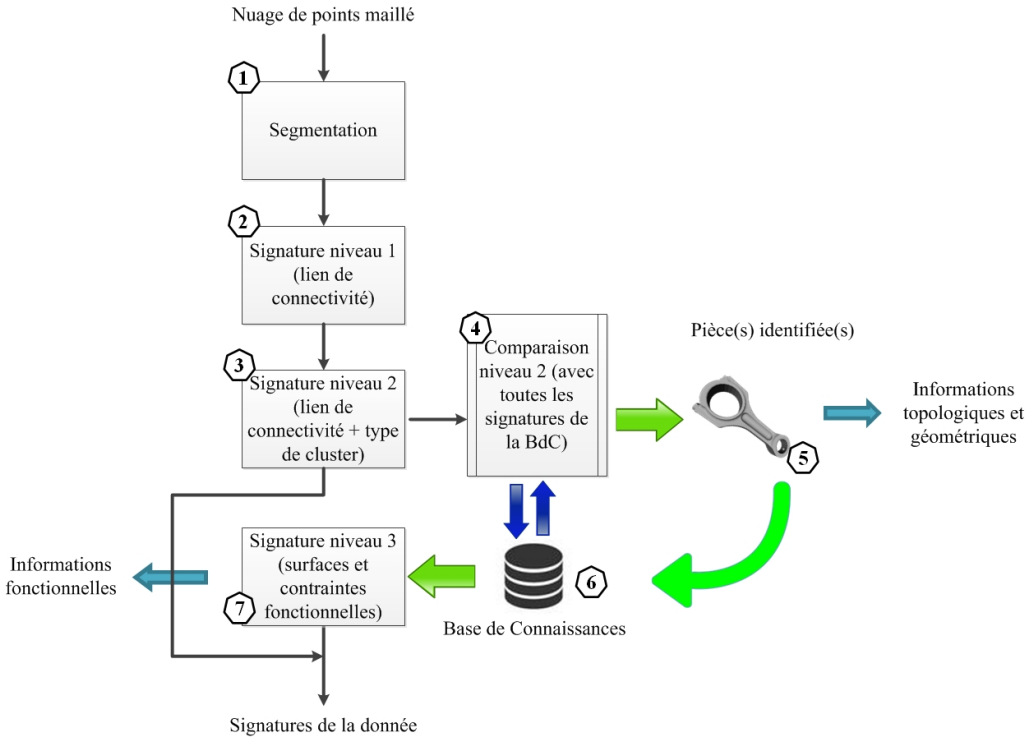


Figure 3.15 – Scénario n°5 : signature de niveaux 1 et 2 puis comparaison de niveau 2 puis signature de niveau 3.

Ce scénario sera utilisé dans un deuxième temps dans la section 4.4 pour l'application à un cas industriel. Son but est d'augmenter le nombre de signatures de la base utilisées pour la comparaison (étape n°4), ce qui augmente également le nombre de résultats obtenus. L'utilisateur possède alors un plus large choix à l'étape n°5 afin de sélectionner les signatures de niveau 3 correspondantes. Comme expliqué au scénario n°2, ce choix (sélection) s'opère en fonction du score de similarité ainsi que du nombre de clusters (nœuds du graphe) et plus particulièrement en fonction du nombre de clusters "typés" dans le *mapping* obtenu pour chaque comparaison de niveau 2 avec une signature de la base. Ceci sera expliqué lors de la mise en application.

La différence avec le scénario n°3 est au niveau de la comparaison de niveau 2 et en particulier de la quantité de signatures utilisée. En effet, la signature de la donnée d'entrée est comparée à toutes les signatures de niveau 2 de la base de connaissances. L'avantage de ce scénario est de pouvoir comparer (pour le niveau 2) des signatures avec un nombre de clusters plus élevé. Cet aspect sera plus largement détaillé dans l'application à un cas industriel (section 4.4).

Nous allons maintenant présenter la mise en œuvre des signatures de niveaux 1 et 2. Nous expliquerons celle concernant le niveau 3 dans la section 3.4 car cette signature dépend des résultats de comparaison.

3.3.1.1 Mise en œuvre du premier niveau

La mise en oeuvre de ce premier niveau a été réalisée à l'aide de CGAL (Computational Geometry Algorithms Library) [CGAL Project, 2015] dans le langage C++. Une description complète de l'algorithme est disponible en Annexe A.2. Cet environnement de développement permet entre autres la lecture et le traitement des données Open Inventor (donnée en sortie d'Efpisoft). Pour cela, de nombreuses bibliothèques en libre accès ont été utilisées dont :

— *3D Polyhedral Surface* : elle permet de construire un polyèdre en trois dimensions composé de sommets, d'arêtes et de facettes (triangles) ainsi que la relation d'incidence entre chacun d'entre eux. De nombreuses fonctions sont ensuite utilisées pour extraire des caractéristiques de chaque polyèdre (continuité de ses facettes par exemple).

— *Principal Component Analysis Reference* : elle fournit des fonctions de calcul afin d'obtenir des informations globales sur un ensemble de données 2D ou 3D. Parmi ces fonctions, le calcul de la boîte englobante orientée est réalisée par $CGAL :: bounding\_box()$. Le calcul du centre de gravité peut être également calculé.

— *2D and 3D Linear Geometry Kernel* : elle donne accès à des fonctions permettant de manipuler les objets 3D et de calculer des distances relatives entre des objets 3D ; ou encore les dimensions et aires de ces objets. Il est également possible de réaliser des transformations affines ainsi que des détections et calculs d'intersections.

Pour pouvoir extraire des informations des fichiers de segmentation, la scène Open Inventor est convertie en polyèdres (chaque cluster étant un polyèdre). Ces derniers sont ensuite traités. Les bibliothèques citées précédemment permettent d'effectuer les calculs suivants sur chaque polyèdre ainsi que sur le polyèdre global de la pièce :

— le centre de gravité ;
— la boîte englobante orientée (aussi appelée Oriented Bounding Box) ;
— la somme des aires des triangles de chaque polyèdre ;
— le critère isopérimétrique : il fait partie des propositions faites dans ce manuscrit et une description complète est apportée dans la section 3.3.2.

En ce qui concerne la construction du graphe de premier niveau, on détermine les relations d'adjacence entre les polyèdres. Les contours des polyèdres correspondant à chaque cluster sont analysés. Pour cela, on observe si les facettes partagent une arête avec une facette appartenant à un polyèdre différent. En projetant cela à un graphe, on obtient alors deux nœuds (chaque nœud correspondant à chaque polyèdre) reliés par une arête (relation d'adjacence).

L'ensemble des données extraites est ensuite mis en forme dans un fichier XML (disponible en annexe A.3). L'architecture du fichier est décrite sur la page suivante.

Nom de la signature
Informations relatives au polyèdre principal (balise <MainPolyhedron>) :
- *centre de gravité*
  - coordonnées suivant x
  - coordonnées suivant y
  - coordonnées suivant z
- *boîte englobante orientée* avec un attribut “type” (BoundingBox par défaut)
  - volume de la BBO
  - coordonnées minimum sur x de la BBO
  - coordonnées maximum sur x de la BBO
  - coordonnées minimum sur y de la BBO
  - coordonnées maximum sur y de la BBO
  - coordonnées minimum sur z de la BBO
  - coordonnées maximum sur z de la BBO
- *somme aire des triangles*
- *valeur critère iso-périmétrique*

Liste des régions (clusters) numérotées de 0 à 11 dans le cas de la bielle figure 3.10 et informations relatives à chacune d’entre elles (balise <RegionList>) :
Numéro de la région et code couleur hexadécimal
- *centre de gravité*
  - coordonnées suivant x
  - etc.
- *Primitive approximée* avec pour attribut le “type” (BoundingBox par défaut)
  - volume de la BBO
  - coordonnées minimum sur x de la BBO
  - etc.
- *somme aire des triangles*
- *valeur critère iso-périmétrique*

Liste des liens de connectivité (balise <Connectivity>) :
Numéro de chaque première région / Numéro de chaque deuxième région en lien.

Une fois le fichier XML généré, il est ensuite comparé avec les signatures de la base de connaissances (voir section 3.4).

3.3.1.2 Mise en œuvre du deuxième niveau

Le but de ce deuxième niveau consiste à typer chaque cluster (région). Pour cela, le XML généré pour le niveau 1 est enrichi. A chaque région, l’attribut “BoundingBox” ajouté par défaut au premier niveau est remplacé par : *plane*, *sphere*, *cylinder* ou *other*.

Dans cette section, nous allons donc exposer la solution développée pour déterminer le type de chaque cluster. L’algorithme proposé a été développé sous MatLab^{TM}. Pour réaliser ce deuxième niveau de signature, deux fichiers d’entrée sont nécessaires (voir figure 3.16) : le fichier XML issu de Efpisoft (conversion de la scène *Open Inventor* en XML) ainsi que la signature de niveau 1 (également sous format XML). Le fichier de sortie est un fichier XML du même nom que la signature de niveau 1 avec la mention “_niv2” apposée juste après.

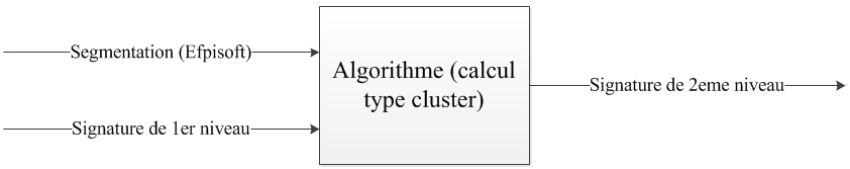


Figure 3.16 – Étape de signature de niveau 2 par graphe.

La description de l'algorithme est présentée dans la figure 3.17. La lecture du fichier XML (Fichier 1) est effectuée par la fonction *xmlread* de Matlab R2014. Elle permet de récupérer les nœuds dans un arbre structuré appelé *Document Object Mode* tree (DOM tree). Les données extraites concernant les régions sont classées dans un tableau REGION à deux dimensions, tel qu'illustré dans la figure 3.17. Chaque ligne correspond à une région (un cluster). La première région est décrite dans cet exemple. La première ligne du tableau Region{1,1} correspond à un triangle composé de trois points : cellule (1,1), cellule (1,2) et cellule (1,3) avec cellule (n° de ligne, n° de colonne). Pour la cellule (1,1), il s'agit du point d'indice 10 du tableau VERTEX. Nous retrouvons alors les coordonnées x, y, et z de ce point aux cellules (10,1), (10,2) et (10,3) respectivement. Le tableau VERTEX contient tous les points du nuage de points.

Les informations liées à chaque région sont ensuite mises de côté afin de définir leur type (boucle i). Trois fonctions sont utilisées pour cela : FittingPlane pour les plans, FittingCylinder pour les cylindres et FittingSphere pour les sphères. Chaque fonction s'inspire de la méthode RANSAC [Fischler et Bolles, 1981] et est décrite par l'algorithme 1.

Algorithme 1 Fitting REGION

En entrée :
Données - un ensemble de points en 3D
modèle - un modèle de surface qui peut être ajusté aux données
t - une valeur seuil pour déterminer si une donnée correspond à un modèle
En sortie :
modèle_possible - les paramètres du modèle qui correspondent le mieux aux données (ou zéro si aucun modèle n'a été trouvé)
pourcentage_points - nombre de points 3D appartenant au modèle
Initialisation :
d - une matrice (1,n)
n ← nombre de points de Données
Algorithme :
modèle_possible ← paramètres du modèle correspondant aux Données {par ex : équation d'un plan}
d = distance (Données, meilleur_modèle) {C'est la distance euclidienne entre chaque point de Données et modèle_possible}
$S = somme(d < t)$ {C'est la somme des valeurs lorsque $d < t$}
pourcentage_points $= S \div n \times 100$

Chaque type (plan, cylindre et sphère) est calculé selon l'algorithme 1 puis les trois résultats *pourcentage_points* sont analysés tel que décrit dans la figure 3.17. Le maximum des trois résultats permet de déterminer le type, si et seulement si, sa valeur est strictement supérieure à 90. Le choix de cette valeur a été arbitraire.

Lorsqu'il y a égalité entre deux pourcentages et qu'ils sont également supérieurs à 90, alors il est impossible de déterminer le type et c'est l'attribut "Autre" qui est affecté à la région.

Concernant le calcul des *modèles*, les fonctions suivantes ont été utilisées :

— pour les plans : la fonction *affine_fit()* disponible en libre accès sur MatLab [Leygue, 2013]. En entrée, c'est une matrice des points sur x, y et z qui est utilisée puis en sortie, nous récupérons les coordonnées de la normale du plan, une matrice (3x2) permettant de créer la base orthogonale du plan ainsi que les coordonnées d'un point P appartenant au plan.

— pour les sphères : la fonction *sphereFit*() disponible sur Matlab a été utilisée [Jennings, 2011]. L'entrée est identique que pour la fonction *affine_fit()* et en sortie, on récupère les coordonnées du centre de la sphère ainsi que son rayon.

— pour les cylindres, un algorithme dédié a été développé dont la description est décrite dans la figure 3.18.

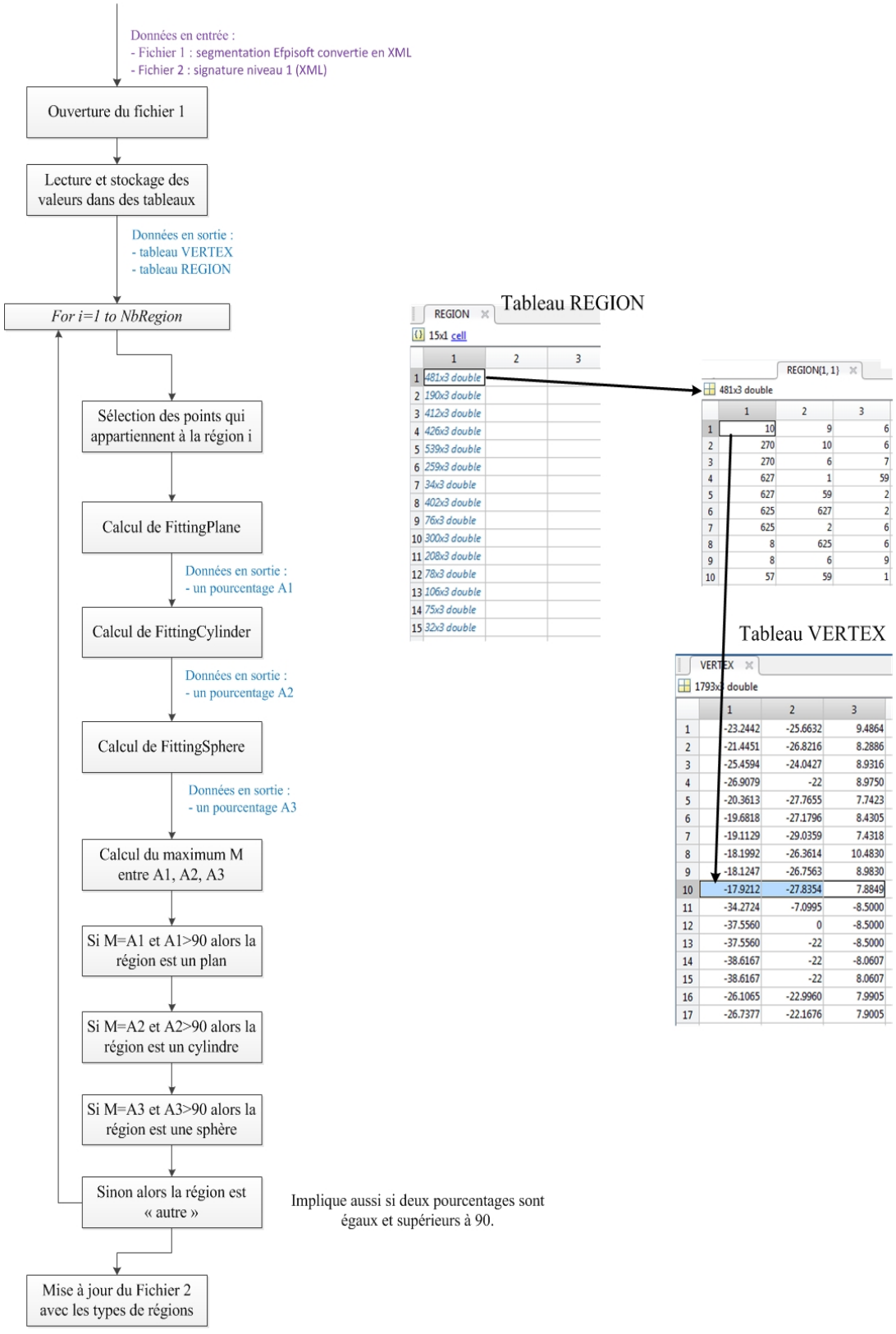


Figure 3.17 – Description de l'algorithme de détection de type de région (cluster).

Le principe de l'algorithme FittingCylinder est de couper le cylindre dans sa latitude grâce à un plan. Les points appartenant à ce plan sont ensuite mis de côté afin d'extraire les caractéristiques du cercle ou de l'arc de cercle formé par la projection de ces points sur le plan. Ainsi, le rayon du cylindre et les coordonnées de son axe sont connus. Ils permettent le calcul de la *distance* (présentée dans l'algorithme 1) de chaque point du nuage par rapport à l'axe du modèle (un cylindre ici).

La première difficulté pour le calcul de *FittingCylinder* réside dans l'orientation du nuage. En effet, si le plan n'est pas bien orienté, il est impossible de calculer le rayon du cylindre. Pour cela, la première étape (voir figure 3.18) consiste à appliquer la fonction *affine_fit()* afin de récupérer un plan qui coupe le cylindre. Cette fonction s'appuie sur le centre de gravité du nuage. Les points appartenant à ce plan sont ensuite isolés en utilisant les caractéristiques du plan : vérification de l'équation du plan par les points de la région et calcul de la distance entre les points et le plan. Seuls les points dont $distance < seuil$ sont stockés dans la matrice Mat1. Le nombre de valeurs de cette matrice est vérifié avec la condition $SiLongueur(Mat) > 3$. Nous considérons qu'il faut minimum 3 points sur ce plan pour ensuite passer à l'étape suivante (TestCercle).

Ensuite, la fonction TestCercle (figure 3.19) permet de calculer le pourcentage de points de la région appartenant au modèle (un cylindre ici) et son principe repose également sur l'algorithme 1. Elle comporte la fonction FitCirclePlane (voir figure 3.20). Cette dernière fonction permet de déterminer précisément le centre du cercle (ou arc du cercle). Pour cela, les points de la matrice Mat1 sont projetés sur le plan de coupe (calcul des produits vectoriels et scalaires). Nous déterminons alors les coordonnées du centre du cylindre grâce à la fonction $ellipse\_fit()$, disponible sur MATLAB [Hendel, 2008]. En sortie de cette fonction, nous récupérons les coordonnées x_p et y_p du centre de l'ellipse (ou cercle) dans le repère lié au plan de coupe. Les coordonnées sont ensuite reconverties dans le repère initial en vue de calculer le rayon du cylindre.

Une fois le $Pourcentage\_points$ calculé avec ce premier plan, nous répétons ces opérations dans les deux autres directions de plan qui correspondent au repère orthogonal. Nous utilisons pour cela les vecteurs v_1 et v_2 du premier modèle calculé (cf. première étape de la figure 3.18). L'ensemble des fonctions précédemment décrites sont ensuite appliquées successivement à ces deux autres plans et c'est le *pourcentage_points* avec la valeur la plus grande qui est conservé (ainsi que son rayon et la direction de son axe).

Remarque Lors des calculs de *Fitting*, les caractéristiques géométriques extraites (ex : coordonnées de la normale d'un plan, direction de l'axe d'un cylindre, rayon d'une sphère, etc.) sont enregistrées dans un fichier .txt structuré qu'il est possible de lire a posteriori. Il sert notamment pour le niveau 3 de signature, lors de l'étape de comparaison des contraintes géométriques. De plus, ces informations pourraient également être utilisées pour reconstruire les modèles CAO paramétrés, notamment afin d'évaluer les paramètres des entités géométriques (exemples : longueur d'extrusion, dimension dans une esquisse, etc.).

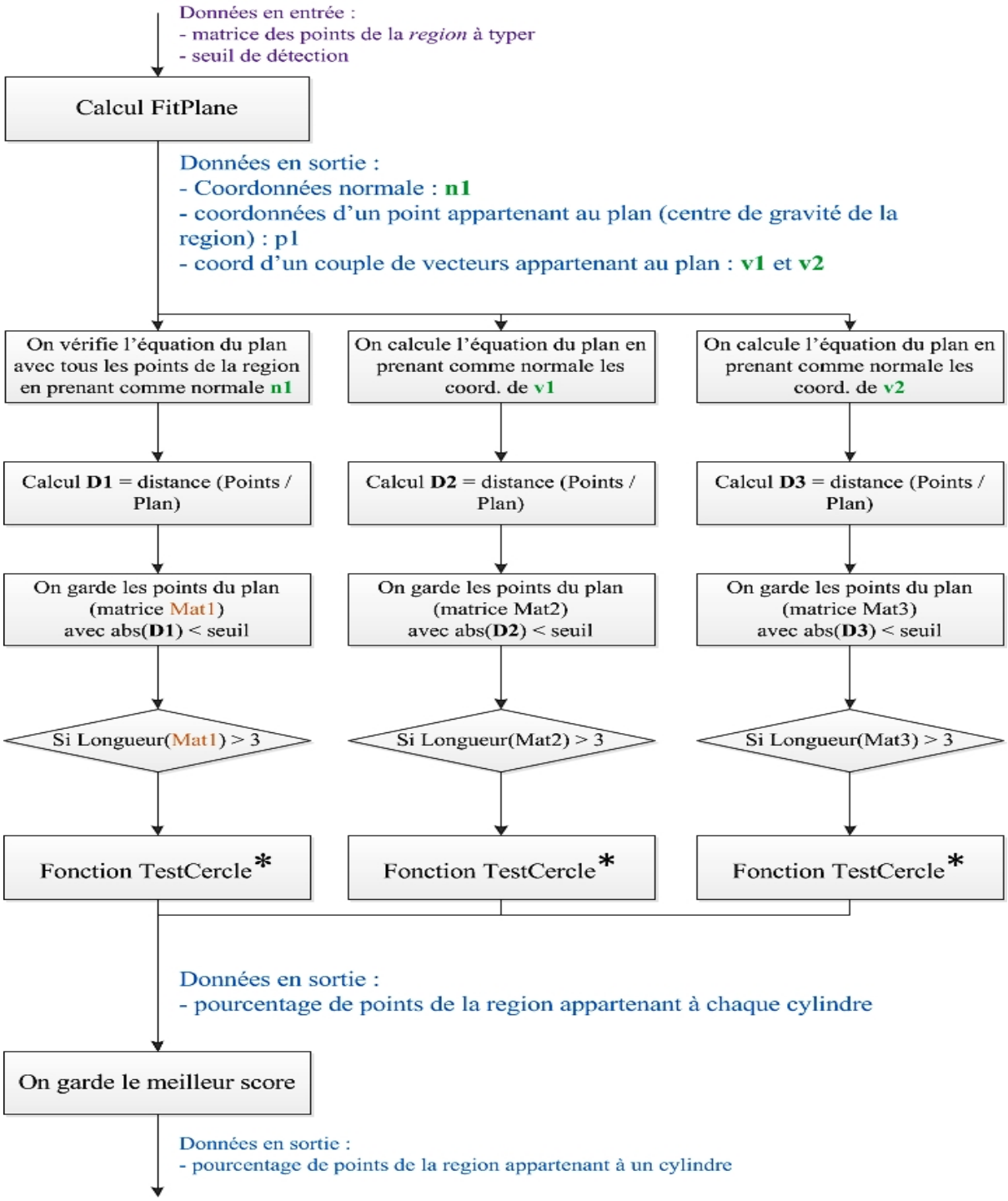


FIGURE 3.18 – Description de l'algorithme FittingCylinder.

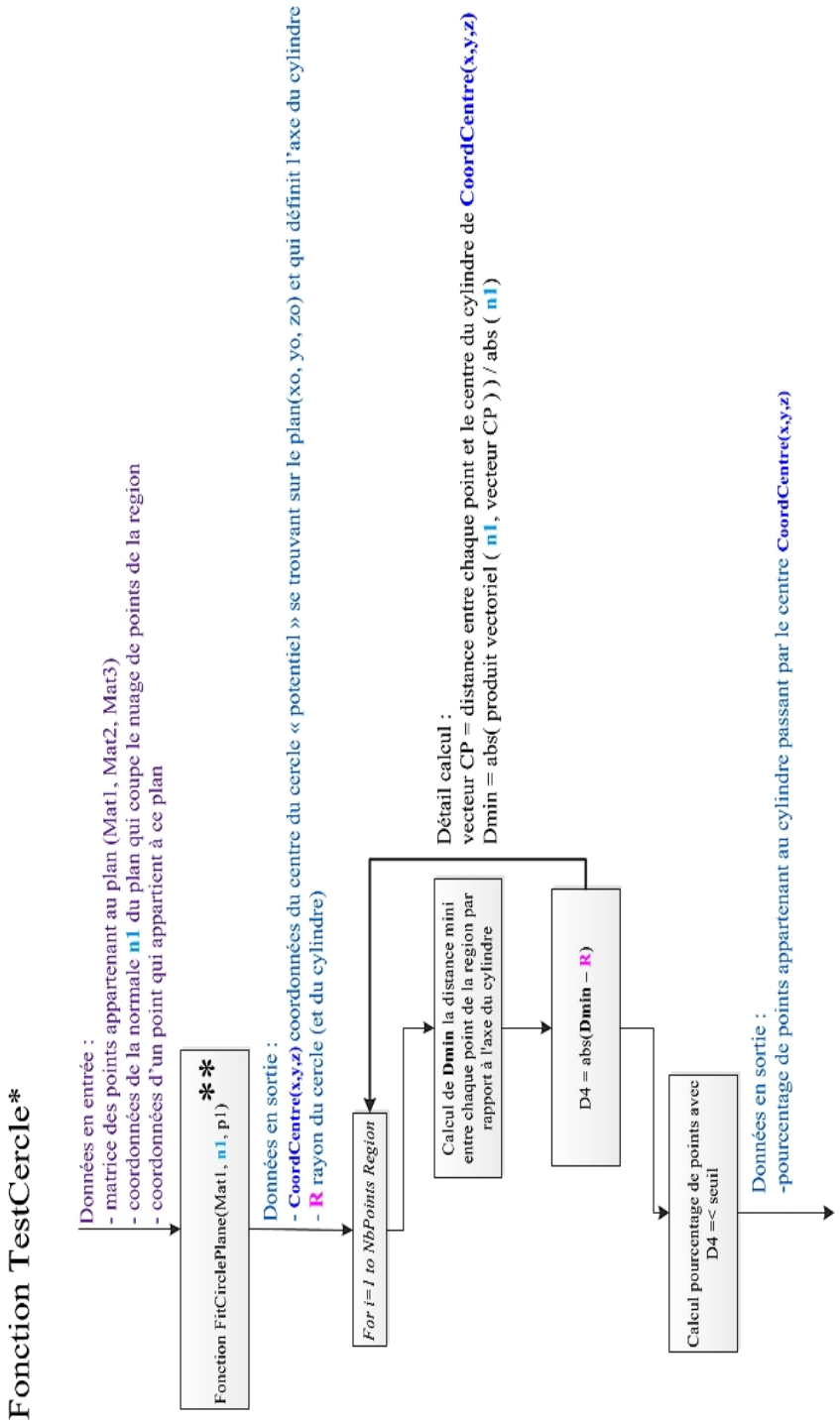


Figure 3.19 – Description de la fonction TestCercle incluse dans FittingCylinder.

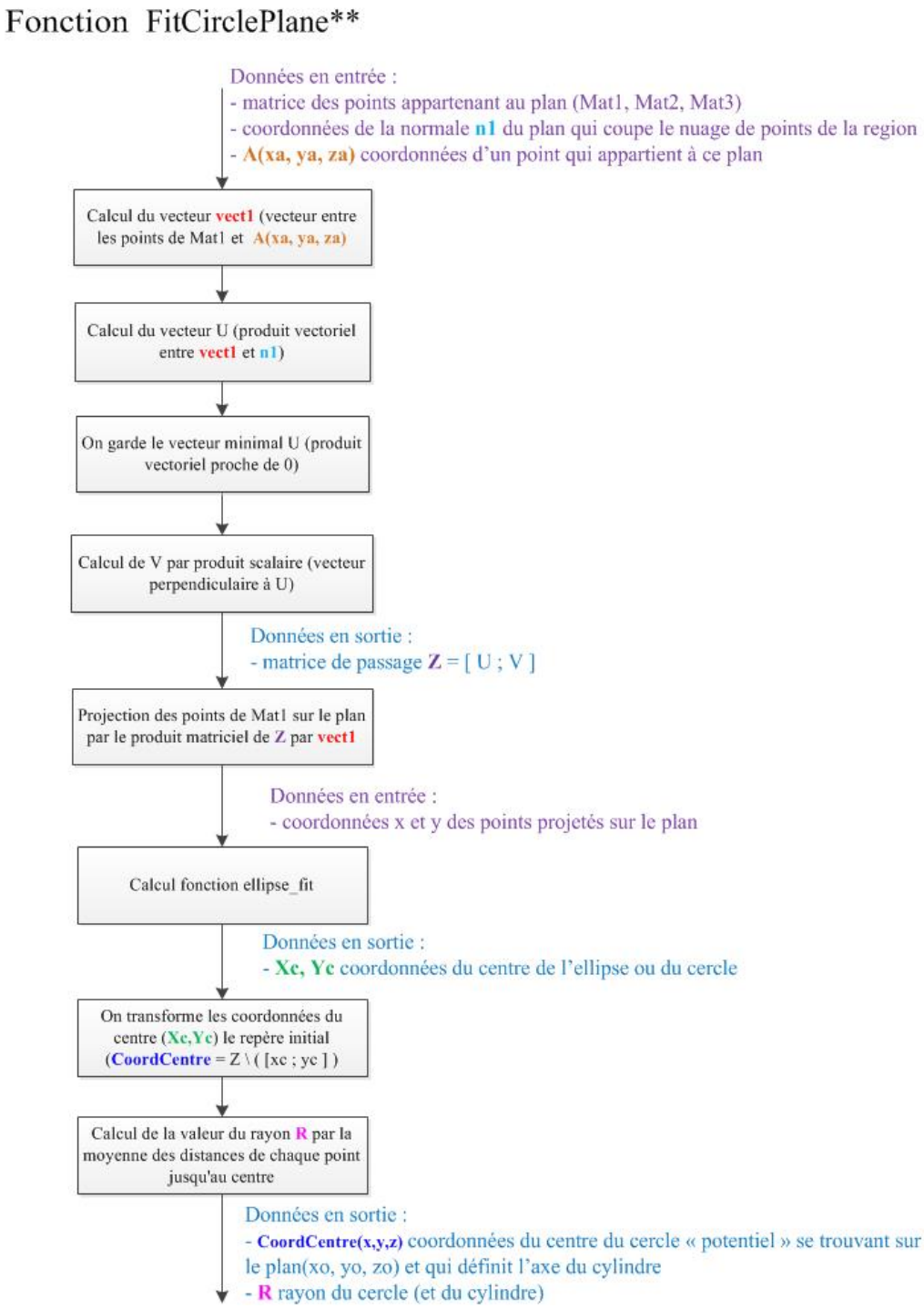


FIGURE 3.20 – Description de la fonction FitCirclePlane incluse dans la fonction TestCercle.

3.3.2 Les signatures par critère global

La plupart des critères globaux de la littérature reposent sur l'inégalité iso-périmétrique qui implique $A^3 \geq 36\pi \times V^2$ dans le cas d'espace continu telle que définie par [Montero et Bribiesca, 2009]. Dans notre cas, le critère choisi est appelé "facteur de compacité" d'une forme et il est défini dans le brevet numéro US 6,169,817 B1 déposé par [Parker *et al.*, 2001]. Le principal avantage repose sur son invariance à l'échelle et à l'orientation de la donnée à signer. En effet, c'est une valeur sans dimension représentant le degré d'une forme à être compacte. Considérons $D \in \Re$ une donnée 3D, $Sa(D)$ la somme des aires de ses faces et $\nu(D)$ le volume de sa boîte englobante orientée. Le facteur de compacité C de D est le nombre :

$$C(D) = \frac{Sa(D)^{1,5}}{\nu(D)} \tag{3.1}$$

Les unités utilisées pour l'aire et le volume sont respectivement en mm^2 et mm^3. La boîte englobante est construite grâce aux axes d'inertie de la donnée. Des plans sont créés par intersection entre chaque direction des axes d'inertie et chaque point extremum appartenant à la donnée 3D.

Voici un exemple de l'application de la formule à un maillage d'un assemblage scanné avec un vilebrequin, une bielle et un piston (voir figure 3.21). Le matériel utilisé pour scanner est un bras laser de marque METRIS avec une précision de ±0,040mm pour la mesure de longueur, situé dans une salle climatisée à une température de 20,5°C. On obtient alors $C(D) = 2,7390$.

FIGURE 3.21 – Exemple d'application du critère iso-périmétrique à un assemblage scanné.

3.3.2.1 Mise en œuvre du critère iso-périmétrique

La mise en œuvre du critère isopérimétrique a été réalisée dans le logiciel Matlab™ version R2014b. La description de l'algorithme est présentée dans la figure 3.22. La fonction *stlread()* développée par [Johnson, 2008] dans MATLAB a été utilisée. Elle permet de lire les fichiers STL binaires uniquement ce qui implique que seuls les fichiers de ce type pourront être signés à l'aide de ce mécanisme. En sortie de cette fonction, nous récupérons deux tableaux : *Faces*(n,3) qui contient n lignes (n le nombre triangles du maillage et 3 colonnes avec les coordonnées x, y, z) et sur chaque ligne nous retrouvons les indices des trois points qui composent chaque triangle. Le deuxième tableau est celui nommé *Vertices*(m,3) qui contient m lignes (m le nombre de points du nuage) et sur chaque ligne, nous avons les coordonnées x, y, z des points. Nous retrouvons une illustration de ces deux tableaux dans la 3.22 avec la cellule (1,1) du tableau *Faces* qui contient le nombre 1 correspondant au point d'indice 1 dans le tableau *Vertices*.

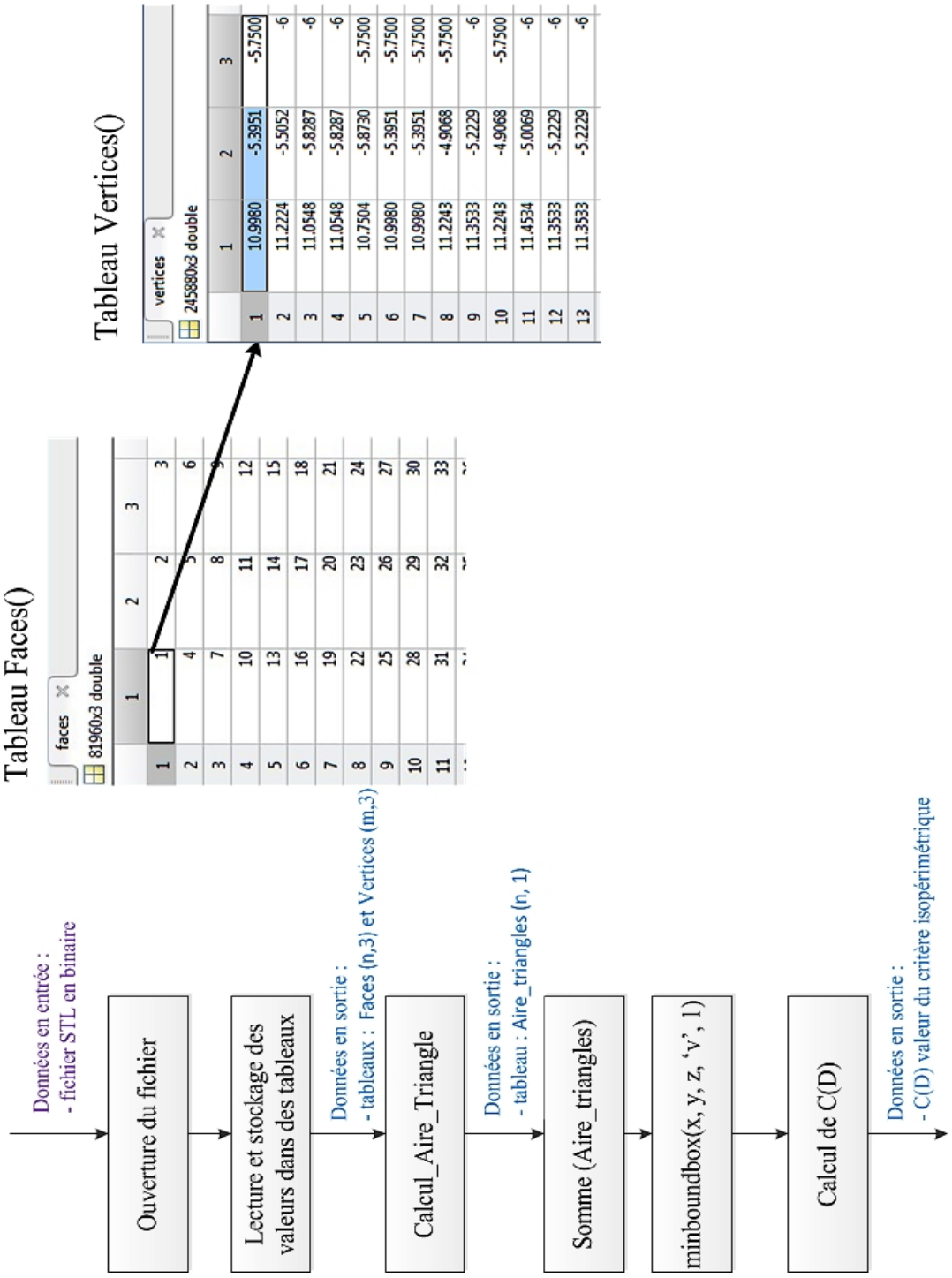


Figure 3.22 – Algorithme de calcul du critère iso-périmétrique sous Matlab^{TM}.

L'aire de tous les triangles est réalisée par la fonction *Calcul_Aire_Triangle()* qui a pour entrées les tableaux *Vertices* et *Faces*. La formule de Héron a été utilisée, elle permet de calculer l'aire d'un triangle en connaissant les longueurs des trois côtés du triangle. Sa définition est la suivante :

$$A = \sqrt{p(p-a)(p-b)(p-c)} \text{ avec } p = \frac{1}{2}(a+b+c) \tag{3.2}$$

avec p le demi-périmètre du triangle ; a, b et c les longueurs des côtés du triangle et A l'aire du triangle.

En sortie de la fonction *Calcul_Aire_Triangle()*, nous avons un tableau Aire_Triangles(n,1) contenant les aires des n triangles. La somme de toutes ces aires est ensuite réalisée.

La fonction *minboundbox()* développée par [Korsawe, 2008] a été implémentée et permet de calculer le volume de la boîte englobante minimale d'un nuage de points. Ses entrées sont les suivantes :

- les coordonnées x, y et z de chaque point du nuage ;
- l'option métrique qui est représentée par une lettre : 'v', 's' ou 'e'. Cette option indique l'utilisation du volume minimal, ou de la surface minimale ou la somme des arêtes minimale comme grandeur à minimiser. L'option par défaut est 'v' et c'est celle qui a été choisie ;
- la méthode utilisée pour le calcul de la boîte englobante : quatre possibilités sont disponibles. Elles sont présentées en détails dans [Korsawe, 2008]. C'est la première méthode qui a été choisie car elle permet d'obtenir un résultat rapide tout en répondant à notre besoin (boîte englobante orientée).

Pour les sorties de cette fonction, nous disposons de :

- une matrice d'inertie (3x3) du nuage de points (*rotmat*) ;
- une matrice *cornerpoints* (8x3) qui correspond aux 8 coordonnées des points qui sont au coin de la boîte ;
- le volume minimal de la boîte ;
- la surface minimale de la boîte ;
- la somme des longueurs des arêtes de la boîte englobante minimale.

Pour le calcul du critère iso-périmétrique, seul le volume minimal est conservé.

La formule 3.1 peut alors être appliquée avec l'aire des triangles du nuage et le volume de la boîte englobante orientée.

Remarque Les deux mécanismes de signature présentés séparément dans cette section 3.3 peuvent être combinés. Suivant le scénario de RE (ou cas industriel), un moteur de workflow pourra être utilisé afin d'identifier quelle(s) signature(s) conviendrai(en)t en fonction du niveau de détails souhaité en sortie de notre processus (maquette numérique avec modèles CAO paramétrés ou seulement une nomenclature). Ce moteur de workflow fait l'objet d'une étude menée par Mohamed Ouamer Ali, doctorant à l'École Centrale de Nantes (partenaire du projet METIS) [Ouamer-Ali *et al.*, 2014].

3.4 Comparaison des signatures avec la base de connaissances

Cette section a pour but de présenter les solutions proposées pour comparer les signatures exposées dans la section précédente. Parmi les deux signatures présentées, seule celle par graphe nécessite une solution pour l'étape de comparaison. En effet, concernant la signature par critère iso-périmétrique, la comparaison consiste à comparer des valeurs et nous avons choisi de ne pas traiter cet aspect-là. Ce sera à l'utilisateur de le faire (manuellement par exemple).

Les solutions choisies pour notre mécanisme de graphe à plusieurs niveaux seront présentées. Nous nous focaliserons sur la partie comparaison de notre mécanisme telle que rappelée dans la figure 3.23.

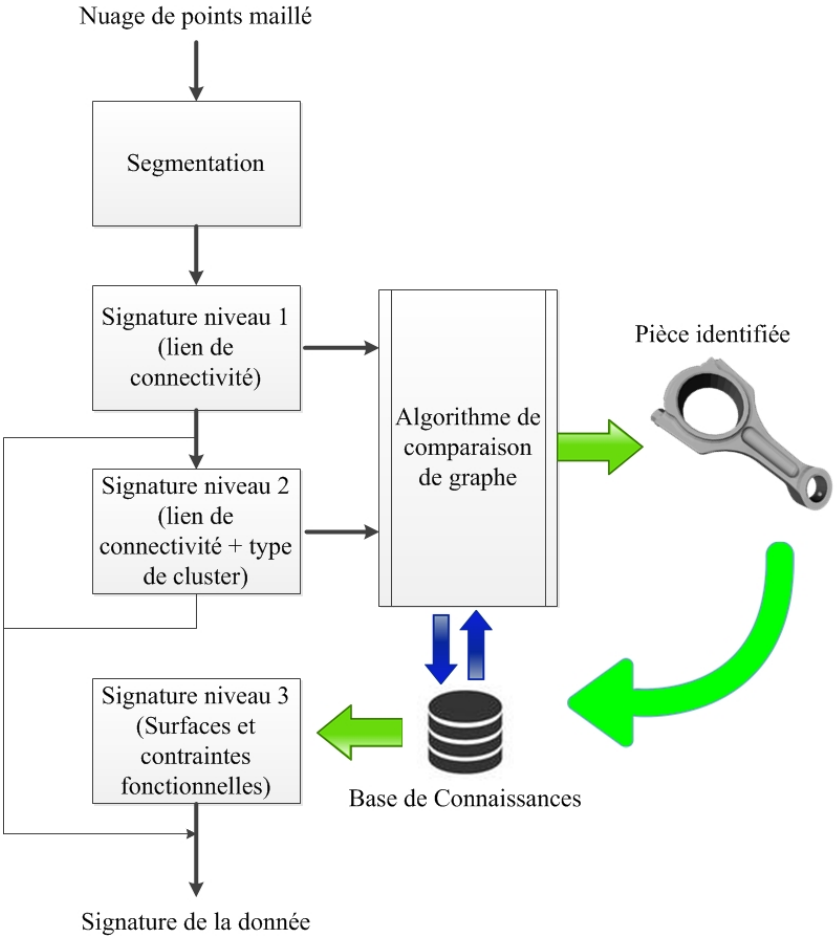


Figure 3.23 – Signature à trois niveaux par graphe.

3.4.1 Comparaison pour le graphe de niveau 1

La solution pour la comparaison de graphe de niveau 1 et 2 a été développée à l'aide de la bibliothèque Boost Graph (BGL). Elle fournit une interface afin de manipuler et créer des graphes. Les éléments sont manipulés grâce une classe de traits [2] : les sommets (aussi appelés nœuds du graphe), les arcs (appelés arêtes dans notre cas car le graphe est non-orienté) et le graphe lui-même.

Parmi les nombreux algorithmes proposés dans la librairie Boost Graph, un nommé *mcgregor common subgraphs* a été utilisé. Il permet de trouver tous les sous-graphes communs entre deux graphes (graphe_1 et graphe_2 par exemple) et affiche la liste des correspondances en sortie.

2. "Une classe de traits est une classe (ou structure) qui associe à un type donné d'autres types (grâce à des typedef) ainsi que des fonctions membres statiques" d'après [Developpez, 2000].

D'après [Rivera, 2009], chaque paire de nœuds non-parcourue appartenant au graphe 1 et au graphe 2 est vérifiée afin de voir s'il est possible d'étendre la vérification au sous-graphe en question. Pour cela, trois étapes sont réalisées (on considère *sommet_1* appartenant au graphe_1 et *sommet_2* au graphe_2) :

1. on vérifie que *sommet_1* et *sommet_2* sont similaires en utilisant le prédicat [3] *sommets_equivalents* ;
2. pour chaque paire de sommets ($sommet1\_existant, sommet2\_existant$) dans le sous-graphe en question, on s'assure qu'il n'existe aucune arête entre *sommet_1* et $sommet1\_existant$ dans le graphe_1 et entre *sommet_2* et $sommet2\_existant$ dans le graphe_2 (en considérant que soit ils existent tous, soit ils sont tous inexistants). Si une ou plusieurs arêtes existent, elles sont vérifiées par équivalence en utilisant le prédicat *sommets_equivalents* ;
3. on s'assure que chaque nouveau sous-graphe est relié par au moins une arête au sous-graphe existant. Cependant cette étape est optionnelle.

Enfin les résultats sont affichés dans un fichier appelé *callback* sous la forme de texte (.txt). Différents types d'informations peuvent être fournis. Dans notre cas, nous retrouvons : les nœuds de chaque graphe comparé ainsi que le *mapping* entre les deux graphes. Il s'agit des sous-graphes maximums entre les deux graphes. Ils sont représentés sous forme de liste où chaque nœud du sous-graphe de l'un est associé au nœud équivalent du sous-graphe de l'autre.

Un exemple complet de cette étape de comparaison est présenté dans la figure 3.24 et le fichier de résultats est consultable dans l'annexe A.4. Nous avons en haut à gauche de la figure, un graphe de 40 nœuds issu d'un maillage (il s'agit de G_1). A droite, nous avons le graphe d'une des signatures de la base de connaissances (G'_1) avec 10 nœuds. Le résultat de la comparaison est alors un ensemble de listes (*mapping*) : seulement quatre d'entre elles sont présentées (le résultat initial offrant environ une centaine de liste de correspondances).

Le premier *mapping* est ensuite sélectionné (choix arbitraire) et nous pouvons apercevoir le sous-graphe commun. En bas de la figure, G_1 (à gauche) et G'_1 (à droite) sont affichés avec en gras les arêtes et nœuds appartenant au *mapping* n°1.

Concernant le score de similarité entre les deux graphes, nous le retrouvons dans le fichier de sortie (voir A.4). Trois facteurs de similarité sont calculés dont voici les expressions :

$$S_{G_1} = \frac{N_{G_{Sub}} \times 100}{N_{G_1}} \tag{3.3}$$

$$S_{G'_1} = \frac{N_{G_{Sub}} \times 100}{N_{G'_1}} \tag{3.4}$$

$$S_{G_1 G'_1} = \frac{2 \times N_{G_{Sub}} \times 100}{N_{G_1} + N_{G'_1}} \tag{3.5}$$

avec S_{G_1} la similarité par rapport au premier graphe ; $S_{G'_1}$ la similarité par rapport au deuxième graphe (G'_1) et $S_{G_1 G'_1}$ la similarité globale (similarité entre les deux graphes) ; N_{G_1} le nombre de nœuds du premier graphe et $N_{G'_1}$ le nombre de nœuds du second graphe. Ces scores sont affichés en pourcentage. Pour chacune des similarités, le nombre de nœuds du plus grand sous-graphe ($N_{G_{Sub}}$) et les nœuds du graphe relatif au calcul de la similarité sont utilisés.

Dans l'exemple de la figure 3.24, on obtient les facteurs de similarités suivants :

$$S_{G_1} = 22\%,\ S_{G'_1} = 90\% \text{ et } S_{G_1 G'_1} = 36\%$$

Nous cherchons à identifier quel composant est identifié dans la donnée d'entrée. Nous garderons alors le score obtenu pour $S_{G'_1}$. Nous pouvons en déduire que la signature du graphe G'_1 est reconnue à 90% dans la donnée d'entrée.

NB : tous les graphes affichés dans ce manuscrit ont été créés manuellement à l'aide du logiciel Visio de Microsoft™ et par lecture du fichier XML de signature.

3. Prédicat : il s'agit d'un énoncé ou d'une proposition qui peut être vraie ou fausse selon ce dont on est en train de parler. Les prédicats permettent de créer les structures conditionnelles (SI, $SINON$) en algorithmique.

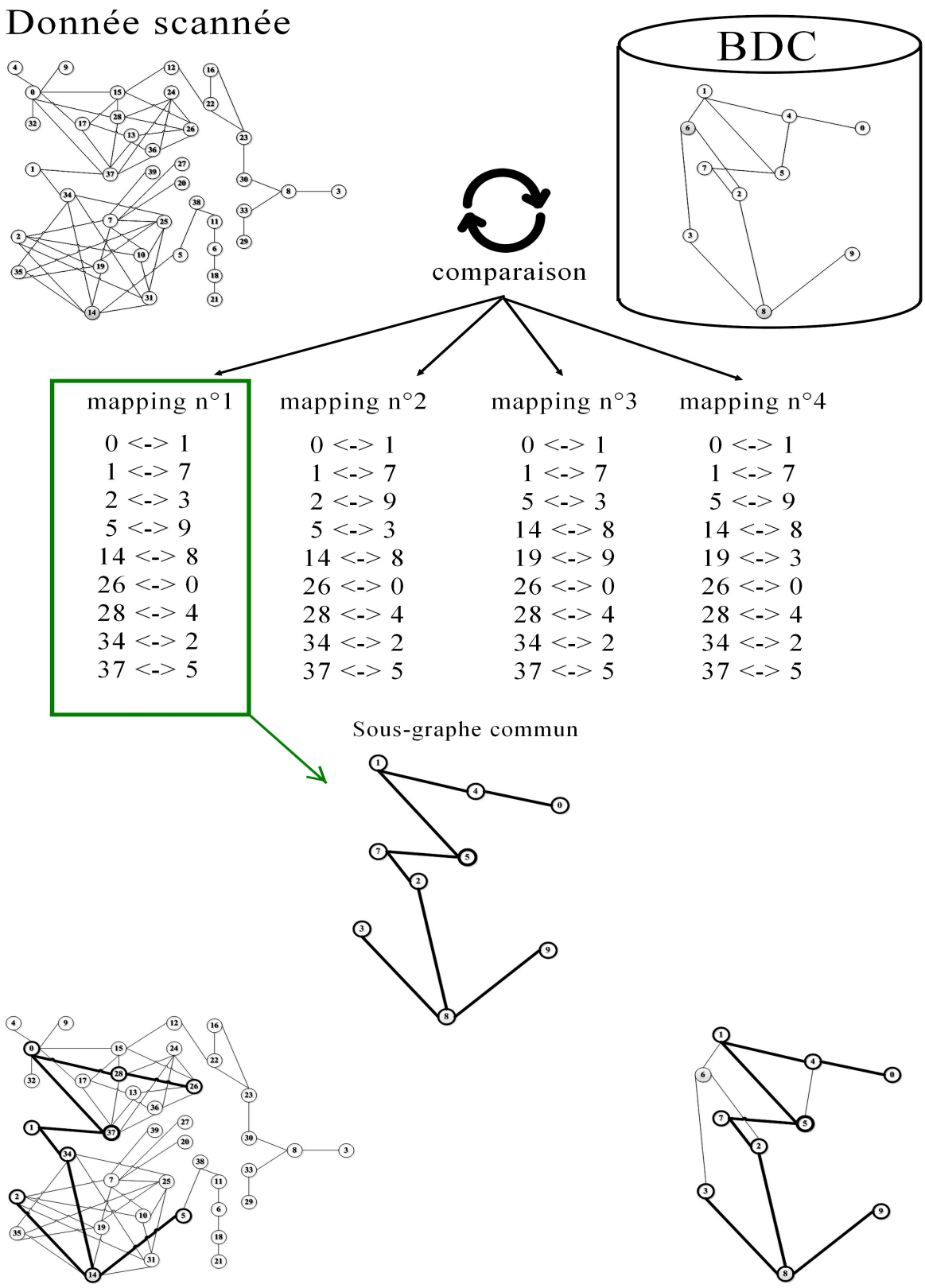


Figure 3.24 – Comparaison de graphe niveau 1 et affichage d'un des *mapping*.

3.4.2 Comparaison pour le graphe de niveau 2

Concernant ce deuxième niveau de comparaison, le principe reste similaire. Cependant un argument supplémentaire est pris en compte : il s'agit du type de région. Il est ajouté comme propriété à chaque nœud du graphe et il entre dans la comparaison des nœuds telle que décrite précédemment.

La figure 3.25 présente un exemple de la comparaison de deuxième niveau. Soit G_2 le graphe de la signature de niveau 2 pour la donnée scannée et G'_2 pour la signature comparée issue de la base de connaissances (G'_2 possédant 25 nœuds). En fonction de leur type, les nœuds ont été colorés : orange pour les cylindres, bleu pour les plans, vert pour les sphères et blanc pour les autres.

Le résultat offre seulement quatre listes de correspondances (*mapping*). La première a été choisie arbitrairement : soit SG_2 le sous-graphe commun correspondant. En bas de la figure, chaque graphe possèdent des nœuds et des arêtes en gras qui correspondent à celles du mapping n°1.

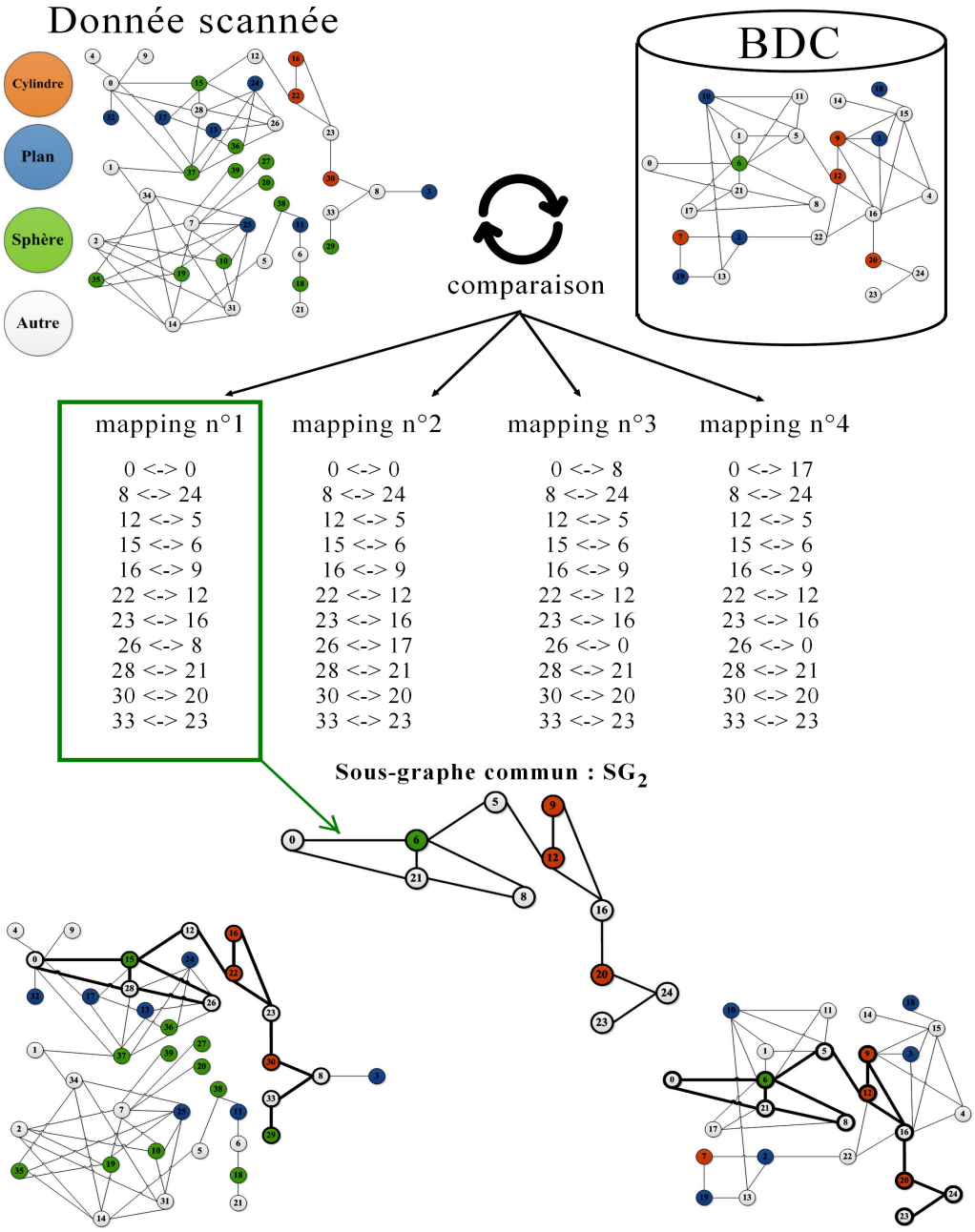


Figure 3.25 – Comparaison de niveau 2 entre deux graphes et affichage d'un des *mapping*.

3.4.3 Comparaison pour le graphe de niveau 3

Comme illustré dans la figure 3.23, le mécanisme de signature découle du résultat de la comparaison de niveau 2. Comme expliqué dans la section 3.3.1, les résultats de la comparaison de niveau 2 sont réutilisés pour la construction de G_3. C'est à partir de là que nous reprenons donc nos explications.

Grâce à la comparaison de niveau 2, nous avons M_2 le meilleur *mapping*, SG_2 le sous-graphe commun et G'_2 le graphe de la signature issue de la base de connaissances. Ce graphe correspond en l'occurrence à celui d'un vilebrequin, ce qui implique (dans notre exemple) qu'un vilebrequin a été identifié dans G_2 suite à la comparaison de niveau 2. Notons que les identifiants des nœuds de SG_2 correspondent à ceux de G'_2 (SG_2 étant un extrait du graphe G'_2 qui correspond au *mapping* n°1 - cf. figure 3.25).

Considérons maintenant G'_3, le graphe de précédence (GP) de ce vilebrequin identifié dans la base (voir graphe de droite sur la figure 3.26). Grâce aux connaissances présentes dans la base, nous avons M_{PG} qui est le *mapping* entre G'_2 et G'_3. Ce *mapping* ne résulte pas d'une comparaison mais d'informations fournies par l'utilisateur en amont du processus de RE. La figure 3.26 illustre G'_2, G'_3 ainsi que le *mapping* M_{PG} qui est réalisé par les flèches en pointillés. Cette mise en correspondance est faite par l'utilisateur à l'aide d'une interface logicielle qui a été développée (présentée en annexe C.1). Une remarque concernant le nœud $Co1$ du GP : il s'agit d'une surface conique. Or ce type de surface n'est pas reconnu dans notre signature, son équivalence est alors de type *Autre* dans le graphe G'_2.

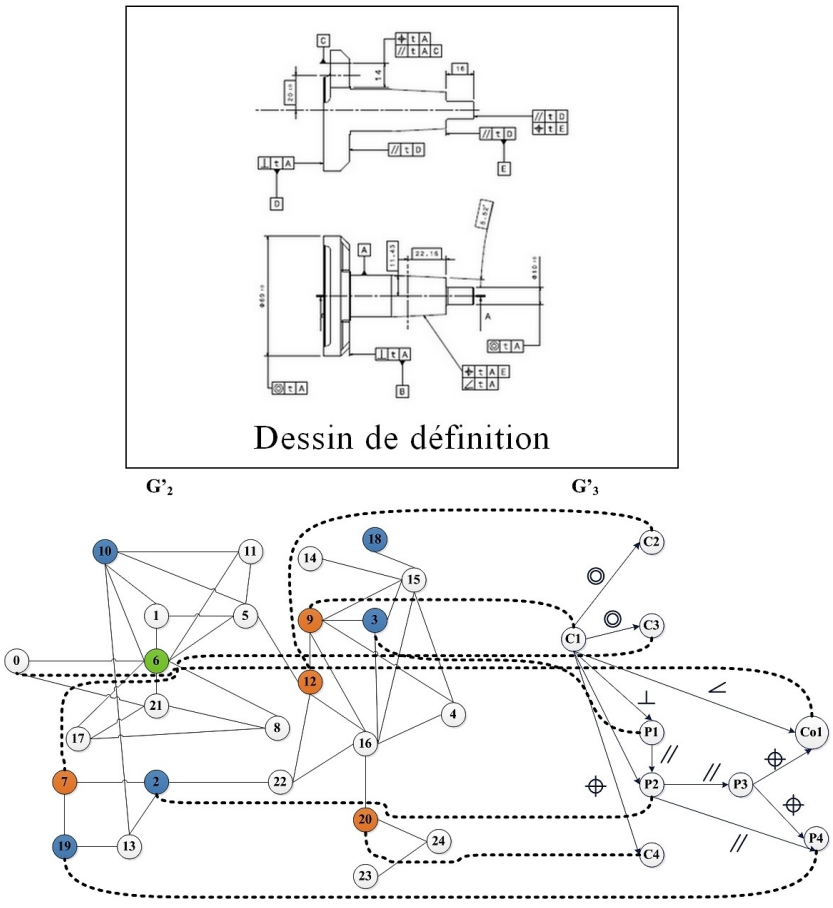


Figure 3.26 – Mise en correspondance entre le graphe d'un vilebrequin en BDC et son graphe de précédence : à gauche le graphe de niveau 2 (les couleurs correspondent à un type de face) et à droite le GP de ce vilebrequin qui découle d'une mise en plan (les nœuds du GP sont typés : P pour plan, C pour cylindre, Co pour cône). Le dessin de définition sert de référence à l'utilisateur pour créer (en amont) le graphe de précédence.

La figure 3.27 représente ainsi le graphe G'_2 fusionné à G'_3 : les arcs du graphe de précédence, représentés par des flèches (graphe orienté) ainsi que les nœuds correspondants à G'_3 ont été surlignés en gras. Soit $GPfus$ ce nouveau graphe présent dans notre base de connaissances. Notons que certains nœuds et arêtes de G'_3 n'ont pu être ajoutés à $GPfus$. En effet, en fonction du nombre de clusters choisi lors de la segmentation, certaines surfaces sont agrégées et sont alors considérées comme *Autre* lors de l'étape de signature. Il est alors impossible de les associer au graphe de précédence.

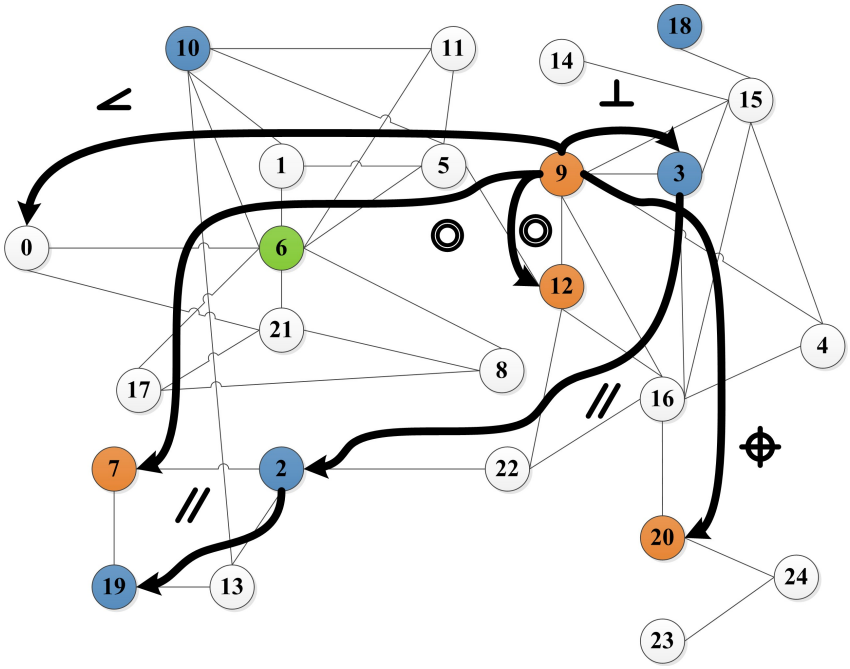


Figure 3.27 – Fusion du graphe de niveau 2 d'un vilebrequin en BDC et de son graphe de précédence.

C'est à partir de ce stade que commence le processus de comparaison de niveau 3. Nous allons alors comparer $GPfus$ et SG_2.

La figure 3.28 illustre les informations présentes en entrée de la comparaison. Nous avons d'une part, le sous-graphe SG_2 et de l'autre le graphe fusionné $GPfus$. Pour l'étape de comparaison de niveau 3, seuls les **nœuds et arêtes en gras** de $GPfus$ seront considérés (les autres nœuds et arêtes sont conservés comme repère pour l'illustration). Cette comparaison est basée sur trois étapes :

1. identifier les nœuds de $GPfus$ qui existent dans SG_2 ;
2. vérifier que les couples [4] de nœuds de $GPfus$ existent dans SG_2 ;
3. dans les couples identifiés, vérifier les contraintes géométriques présentes dans SG_2 (étiquettes sur les arcs).

L'algorithme 2 décrit l'exécution de la comparaison. Les valeurs de chaque graphe sont mises dans un tableau et nous les comparerons. Lorsqu'un couple de nœuds est possible, nous vérifions alors la contrainte géométrique avec la fonction $Verifier\_Contrainte\_Geometrique()$ qui renvoie $True$ ou $False$ (vrai ou faux).

Cette dernière s'appuie sur les informations extraites de chaque région lors du processus de signature de niveau 2. Ces informations sont stockées dans un fichier .txt que nous venons lire et réutiliser afin de réaliser la comparaison géométrique.

4. Couple : ce sont deux nœuds reliés par au moins un arc.

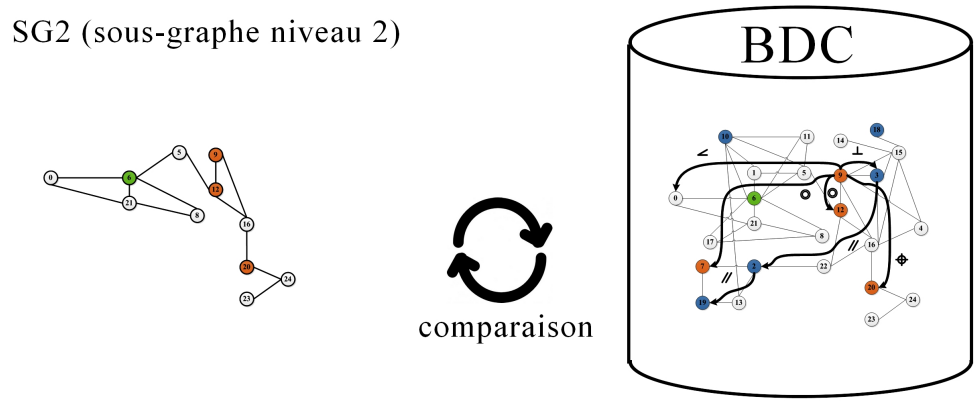


Figure 3.28 – Comparaison de niveau 3 entre le sous-graphe SG_2 et $GPfus$ le graphe associé au GP.

Selon la contrainte, différentes opérations sont alors réalisées :

— pour vérifier que deux plans sont parallèles : calcul de la colinéarité des normales de chaque plan ;
— pour vérifier que deux cylindres sont parallèles : calcul de la colinéarité des directions de leur axe respectif ;
— pour vérifier que deux plans sont perpendiculaires : calcul du produit scalaire des normales de chaque plan ;
— pour vérifier que deux cylindres sont coaxiaux : calcul de la colinéarité des directions de leur axe respectif puis calcul de la distance entre les centres des cylindres (position de chaque axe).

Pour chacun de ces calculs, les valeurs obtenues sont comparées à une valeur seuil (proche de zéro) définie par l'utilisateur.

Algorithme 2 Comparaison niveau 3

En entrée :
Tab_GPFus - un tableau (i,3) contenant i couples de nœuds et 3 colonnes (nœud1, nœud2, contrainte géométrique) correspondant au graphe $GPfus$
Tab_SG2 - un tableau (j, 1) contenant j nœuds correspondant au sous-graphe SG_2
En sortie :
facteur de similarité entre les deux graphes comparés
Initialisation :
Couple_Possible - un tableau (n, 2) contenant n couples de nœuds possibles
Tab_GraphNiv3 - le tableau (k,3) correspondant au graphe de niveau 3 (du vilebrequin)
count = 1 {compteur}
Algorithme :

```
for i = 1 to longueur(Tab_GPFus) do
  for j = 1 to longeur(Tab_SG2) do
    if Tab_GPFus(i, 1) == Tab_SG2(j) then
      Nœud_Possible(j,1) = Tab_GPFus(i,1).valeur
      —
      for k = 1 to longueur(Tab_GPFus) do
        for l = 1 to longeur(Tab_SG2) do
          if Tab_GPFus(k, 2) == Tab_SG2(l) then
            Nœud_Possible(k,2) = Tab_GPFus(k,2).valeur
            —
            if Vérifier_Contrainte_Géométrique() is TRUE then
              Tab_GraphNiv3(i,1)= Tab_GPFus(i,1).valeur
              Tab_GraphNiv3(k,2)= Tab_GPFus(k,2).valeur
              Tab_GraphNiv3(count,3)= Tab_GPFus(i,3).valeur
              count = count +1
              —
            end if
          end if
          —
          l=l+1
        end for
        k=k+1
      end for
      —
    end if
    j=j+1
  end for
  i=i+1
end for
```

Écriture du fichier XML de niveau 3 d'après le tableau Tab_GraphNiv3()

Pour reprendre l'exemple de la figure 3.28, nous allons lui appliquer l'algorithme 2. La figure 3.29 illustre les tableaux lors de l'initialisation.

Tab_SG2( )

Nœud 1
0
5
6
8
9
12
16
20
21
23
24

Tab_GPFus( )

Nœud 1	Nœud 2	Contrainte géométrique
9	0	Inclinaison
9	3	Perpendicularité
9	7	Coaxialité
9	12	Coaxialité
9	20	Localisation
3	2	Parallélisme
2	19	Parallélisme

Couple_Possible( )

Nœud 1	Nœud 2

Tab_GraphNiv3( )

Nœud 1	Nœud 2	Contrainte géométrique

FIGURE 3.29 – Initialisation de l'algorithme et affichage des tableaux des graphes.

Nous commençons par parcourir le tableau Tab_GPFus :

- **1ère itération :** nous comparons la première valeur du Nœud 1 (9) avec les valeurs du tableau Tab_SG2. Ce dernier est parcouru ligne par ligne.
  - Si la valeur trouvée à la ligne de Tab_SG2 est égale à 9 alors on complète $Couple\_Possible(1,1)$ par la valeur 9. On compare maintenant la première valeur de Nœud 2 de Tab_GPFus (0) avec les valeurs du tableau Tab_SG2.
    - Si la valeur trouvée à la ligne de Tab_SG2 est égale à 0 alors on complète $Couple\_Possible(1,2)$ par la valeur 0. Puis on vérifie si la contrainte géométrique comprise dans $Tab\_GPFus(1,3)$ (i.e. *Inclinaison*) est respectée entre les nœuds 0 et 9 du graphe SG_2.
      - si *Vérifier_Contrainte_Géométrique()* est VRAIE alors : $Tab\_GraphNiv3(1,1) = 9$ et $Tab\_GraphNiv3(1,2) = 0$ et $Tab\_GraphNiv3(1,3) = Inclinaison$
      - sinon on passe à la ligne suivante du tableau Tab_SG2
    - sinon on passe à la ligne suivante du tableau Tab_GPFus
  - sinon on passe à la ligne suivante du tableau Tab_GPFus.
- **2ème itération :** de la même manière, nous allons chercher si les nœuds 3 et 9 du tableau Tab_GPFus existent dans Tab_SG2. Dans notre exemple, le nœud 3 n'existe pas dans Tab_SG2. Ce couple de nœuds ne peut pas alors être comptabilisé.
- etc. jusqu'à la septième ligne du tableau Tab_GPFus.

Les différentes itérations et les résultats obtenus sont présentés dans la figure 3.30. Le calcul de vérification des contraintes géométriques n'est pas présenté dans ces travaux et fera partie des perspectives. Nous faisons donc l'hypothèse qu'elles ont été vérifiées et validées.

Couple_Possible( )

	Nœud 1	Nœud 2
1ère itération	9	0
2e itération	9	Ø
3e itération	9	Ø
4e itération	9	12
5e itération	9	20
6e itération	Ø	Ø
7e itération	Ø	Ø

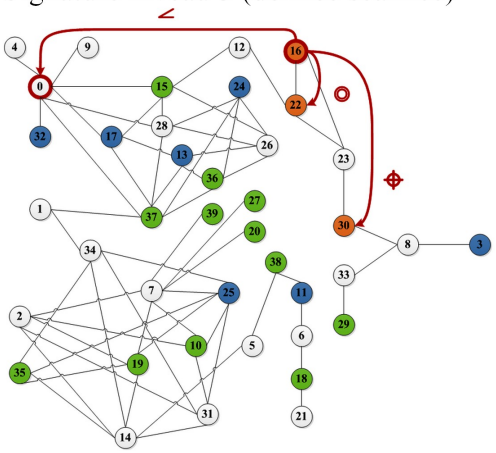
Tab_GraphNiv3( )

Nœud 1	Nœud 2	Contrainte géométrique
9	0	Inclinaison
9	12	Coaxialité
9	20	Localisation

FIGURE 3.30 – Illustration de l'algorithme de comparaison avec l'exemple de la figure 3.28 et affichage des couples possibles obtenus pour chaque itération de l'algorithme.

La dernière étape consiste à générer le graphe G_3 à l'aide du tableau $Tab\_GraphNiv3$ en convertissant les nœuds de Tab_SG2 grâce au mapping M_2. Le graphe généré est alors la signature de niveau 3 de notre donnée scannée. Cette signature est ensuite conservée pour la suite du processus de RE (pour la reconstruction du modèle CAO en particulier). La figure 3.31 illustre les signatures de niveaux 1, 2 et 3 où le graphe de précédence est représenté avec des nœuds et des arêtes surlignés.

Signature niveau 3 (donnée scannée)



Mapping niveau 2

0 <-> 0
8 <-> 24
12 <-> 5
15 <-> 6
16 <-> 9
22 <-> 12
23 <-> 16
26 <-> 8
28 <-> 21
30 <-> 20
33 <-> 23

Tab_GraphNiv3()

Nœud 1	Nœud 2	Contrainte géométrique
9	0	Inclinaison
9	12	Coaxialité
9	20	Localisation

FIGURE 3.31 – Signature de niveau 3 de la donnée scannée : à gauche, le graphe de niveau 3 de la donnée scannée, au milieu la liste de correspondances (*mapping*) entre le graphe de la donnée en BDC et le graphe de la donnée scannée et à droite, le tableau de résultats de la comparaison de niveau 3.

3.4.4 Mise en œuvre de la comparaison de graphe

Dans cette section, nous citerons les principales fonctions utilisées afin de réaliser l'algorithme de comparaison qui est commun aux trois niveaux de graphe.

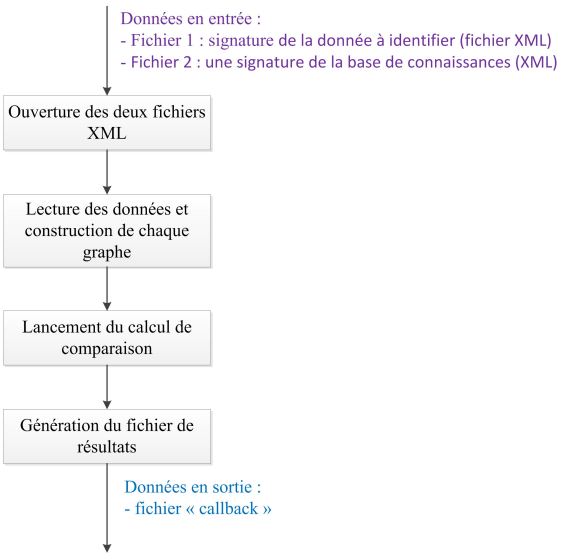


FIGURE 3.32 – Structure de l'algorithme de comparaison de niveau 1.

La figure 3.32 présente les différentes étapes de cet algorithme. Pour ce qui concerne la lecture des données afin de construire les graphes, les fonctions suivantes ont été utilisées :

— $Graphgraph\_simple1 = extractGraphFromXMLFile(XmlFile1, withRegionType, withPrecedenceGraph)$ et $Graphgraph\_simple2 = extractGraphFromXMLFile(XmlFile2, withRegionType, withPrecedenceGraph)$ permettent d'importer les deux graphes. Les arguments $withRegionType$ et $withPrecedenceGraph$ servent respectivement aux comparaisons de niveau 2 et 3.

— $boost :: threadt1(search\_mcgregor\_common\_subgraphs, ci)$; permet de lancer le calcul de comparaison via le *thread* [5] ;
— une horloge a été ajoutée dans le code et permet d'interrompre la comparaison en cours et de passer à la suivante (le cas échéant) ;
— enfin la fonction *print_callback* permet la génération du résultat dans un fichier .log (texte).

Concernant le calcul des facteurs de similarité, un compteur de noeuds est utilisé.

5. Thread : c'est une commande qui lance un processus (ici l'algorithme de comparaison *mcgregor_common_subgraphs*).

3.5 Conclusion sur la proposition

Dans ce chapitre, nous avons présenté les différentes solutions afin de répondre à notre problématique. La méthodologie HDI-RE a été proposée et pour chaque étape, différentes solutions ont été apportées :

- pour la segmentation : l'utilisation du logiciel Efpisoft a permis de segmenter les données issues de nuage de points. Cette application est basée sur une segmentation par clusterisation hiérarchique.

- pour la signature : une première solution portant sur un critère iso-périmétrique a été choisie. Elle permet d'obtenir une caractéristique globale de l'assemblage à identifier. Puis une deuxième signature basée sur les graphes a été proposée. Cette dernière est à trois niveaux : le premier niveau est un graphe de connectivité, le deuxième est un graphe de connectivité avec le type de nœuds (attribut sur les nœuds) et le troisième niveau est un graphe de précédence contenant des contraintes géométriques entre les nœuds déjà typés (attribut sur les nœuds et les arêtes). Ces signatures permettent de décrire la donnée grâce à trois niveaux d'information : le niveau global, le niveau topologique et géométrique et le niveau fonctionnel.

- pour la comparaison : la bibliothèque Boost Graph ainsi que l'algorithme Mc_gregor_subgraph ont été choisis afin de comparer les signatures par graphe.

CHAPITRE 4

Expérimentation de la proposition et validation sur un cas industriel

"La richesse d'un concept scientifique se mesure à sa puissance de déformation."

- Gaston Bachelard

Introduction

Ce dernier chapitre est une discussion sur la validité industrielle et la faisabilité de HDI-RE. Dans un premier temps, nous présenterons différents tests réalisés sur chaque solution apportée pour chaque étape de notre méthodologie ainsi que les résultats correspondants. Puis dans un deuxième temps, nous appliquerons notre méthodologie à un cas industriel.

Le chapitre est composé des sections suivantes :

- section 4.1 : nous étudierons l'influence du nombre de clusters choisi lors de l'étape de segmentation sur notre méthodologie HDI-RE ;
- section 4.2 : nous testerons nos algorithmes de détection (*FittingPlane*, *FittingCylinder* et *FittingSphere*) sur des données scannées de référence. Puis dans la sous-section 4.2.2, nous évaluerons le critère iso-périmétrique sur plusieurs assemblages mécaniques ;
- section 4.3 : nous examinerons les résultats obtenus pour la comparaison de graphes pour les niveaux 1 et 2 ;
- section 4.4 : nous appliquerons l'ensemble de notre méthodologie à un assemblage simple (piston, bielle) issu de données industrielles.

4.1 Expérimentation de la segmentation

La première partie des tests concerne la segmentation des données 3D (maillages issus de nuages de points) réalisée grâce au logiciel Efpisoft. Comme expliqué dans la section 3.2, la donnée est découpée en clusters (agrégation de triangles) dont le nombre est ajusté par l'utilisateur. Cette opération a un impact sur l'étape de signature par graphe et en particulier au niveau 2 où chaque cluster est typé (plan, cylindre, sphère ou autre). En effet, suivant leur nombre, les clusters ne s'ajustent à aucune surface canonique de référence définie dans nos travaux (plan, cylindre ou sphère) et le type *Autre* leur est associé. C'est notamment le cas lorsque le nombre de clusters est faible et qu'un cluster regroupe plusieurs faces possibles de la pièce à identifier. Un exemple de ceci est présenté dans la figure 4.1. Nous remarquons que le cluster n°1 est associé à une surface cylindrique ainsi qu'à une surface plane. Pour le cluster n°2, plusieurs cylindres sont fusionnés alors qu'ils devraient être distincts. Il en est de même avec le cluster n°3. Ces trois clusters ne s'ajusteront avec aucune surface canonique de référence et le type *Autre* leur sera attribué.

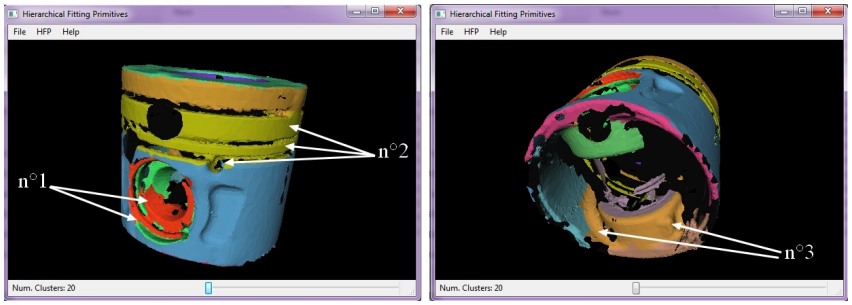


Figure 4.1 – Segmentation d'un piston sous Efpisoft avec 20 clusters.

Nous avons donc choisi d'étudier l'influence du nombre de clusters sur la signature qui en découle et en particulier par rapport au nombre de surfaces de référence ajustées et "non-ajustées" (de type *Autre*).

Principe du test Pour réaliser ces tests, nous avons réalisé les opérations suivantes :

1. segmentation de la donnée 3D en définissant un nombre précis de clusters ;
2. génération de la signature de niveau 1 ;
3. génération de la signature de niveau 2 sous MATLAB ;
4. décompte automatique du nombre de surfaces ajustées, non-ajustées par rapport au nombre total de clusters défini.

Le test a été appliqué à trois pièces différentes : une bielle, un piston et un vilebrequin que nous pouvons apercevoir sur la figure 4.2. Ces pièces ont été fournies par l'IFPEN (Institut Français du Pétrole et des Énergies Nouvelles), un des partenaires du projet sur lequel reposent ces travaux. Il s'agit de pièces de véhicule automobile scannées à l'aide d'un *HandySCAN 3D*™. Les trous sur les nuages sont dus à des pastilles collées sur les pièces.

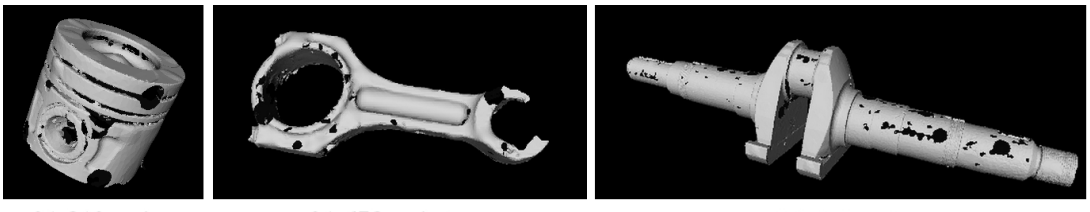


Figure 4.2 – Ensemble des données scannées testées.

Variables Le nombre de clusters évolue de 5 en 5 en commençant par 5 et jusqu'à 100 clusters.

Pour l'étape de signature de niveau 2, la valeur du seuil de détection des formes canoniques varie également. Comme expliqué dans la section 3.3.1.2, le seuil (de détection) permet de définir si un point du maillage ou un ensemble de points appartient à un modèle défini (plan, sphère ou cylindre). La valeur de celui-ci n'est pas encore définie à cette étape de l'expérimentation de notre méthodologie. C'est pourquoi, les tests présentés dans cette section sont réalisés en faisant varier cette valeur sur les trois algorithmes (*FittingPlane*, *FittingCylinder*, *FittingSphere*). La section 4.2.1 quant à elle, présentera des tests pour définir une valeur fixe pour chaque type de surface canonique.

La figure 4.3 illustre les deux variables dont il est question dans ce test : nous retrouvons un exemple de variation du nombre de clusters (5, 35, 50, 70 et 100 clusters). Puis en-dessous, dans la figure (a), nous pouvons observer un nuage de points (cluster isolé) orienté dans l'espace (x, y, z). Un modèle représentant un plan (en rouge) est ajusté à cet ensemble de points. On calcule alors la distance entre chaque point du nuage et le modèle. Puis nous définissons une valeur de seuil qui inclut ou exclut le point par rapport au modèle (avec $|distance| < seuil$). Nous avons effectué trois tests : avec un seuil de 1 mm, 1,5 mm et 2 mm. Sur l'exemple de la figure 4.3 (b), on considère alors que les points inclus entre les deux lignes épaisses (en vert) appartiennent au modèle ($|distance| < 1mm$).

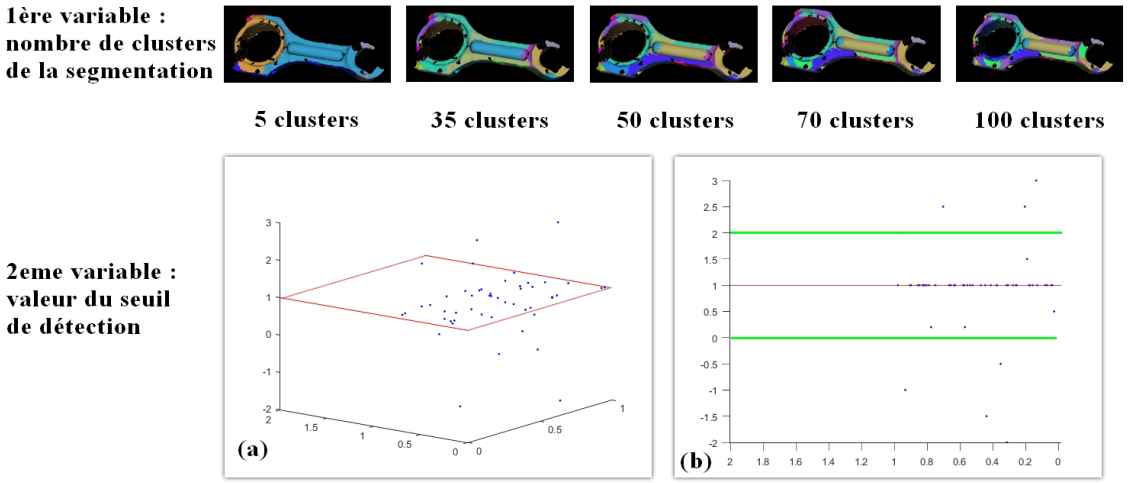


FIGURE 4.3 – Variables pour les tests expérimentaux sous Efpisoft

Les résultats de ces tests sont présentés dans les tableaux suivants :
— Tableau 4.1 pour la bielle ;
— Tableau 4.2 pour le piston ;
— Tableau 4.3 pour le vilebrequin ;
avec en colonne, le nombre de clusters (de 5 à 100) et en ligne la valeur du seuil (1, 1,5 et 2 mm). A chaque croisement ligne-colonne, nous trouvons le pourcentage de surfaces ajustées par rapport au nombre total de clusters à typer. En-dessous de chaque tableau, nous retrouvons un graphique avec en abscisse le nombre de clusters du maillage et en ordonnée, le pourcentage de surfaces ajustées obtenu.

Pour les trois tableaux, nous observons que le nombre de surfaces ajustées augmente conjointement avec le nombre de clusters de la segmentation. Nous remarquons également que la valeur du seuil joue le rôle de facteur d'échelle : plus la valeur est grande, plus le nombre de surfaces ajustées est élevé. Nous notons cependant qu'à partir d'un certain nombre de clusters (par exemple 55-60 clusters pour la bielle avec un seuil de 2mm), la valeur du pourcentage stagne.

Pourcentage de surfaces ajustées par rapport au nombre de clusters	Nb clusters	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Seuil (en mm)	1	0	0	0	0	0	7	11	20	22	30	31	37	41,5	47,1	49,3	52,5	54,1	60	65,3	69
	1,5	0	0	0	0	12	13	29	45	49	58	62	67	71	74	80	81	82	84	86	88
	2	0	10	13	30	48	53	69	73	82	86	91	90	91	93	93	94	95	96	96	96

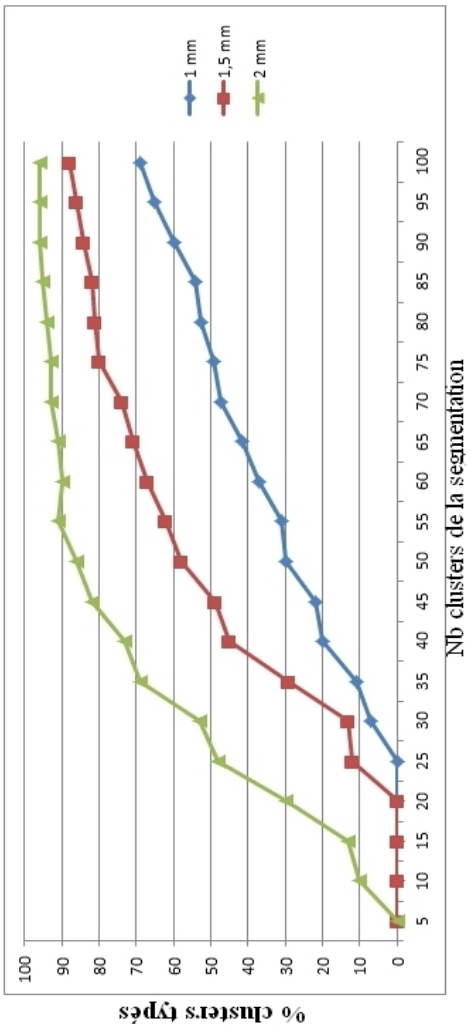


Tableau 4.1 – Évolution du nombre de surfaces ajustées par rapport au nombre de clusters pour une bielle scannée.

Pourcentage de surfaces ajustées par rapport au nombre de clusters	Nb clusters	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Seuil (en mm)	1	0	20	13	25	20	27	26	35	38	38	38	43	46	49	51	55	59	62	63	67
	1,5	0	20	13	25	40	53	63	70	76	76	78	82	85	86	88	89	92	93	95	96
	2	0	30	20	40	60	77	86	93	96	96	95	95	95	96	97	98	98	98	98	98

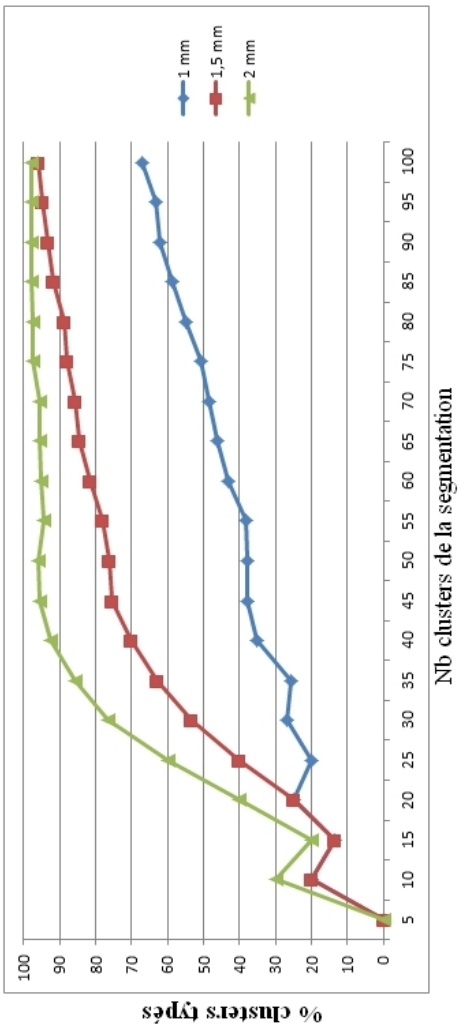


Tableau 4.2 – Évolution du nombre de surfaces ajustées par rapport au nombre de clusters pour un piston scanné.

Pourcentage de surfaces ajustées par rapport au nombre de clusters	Nb clusters	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Seuil (en mm)	1	0	0	0	0	4	13	23	30	33	36	38	48	49	54	56	61	64	64	67	68
	1,5	0	0	7	15	20	27	34	43	49	52	56	60	65	70	72	74	74	76	78	79
	2	0	10	13	20	32	40	46	53	58	62	64	65	71	77	79	81	82	82	83	84

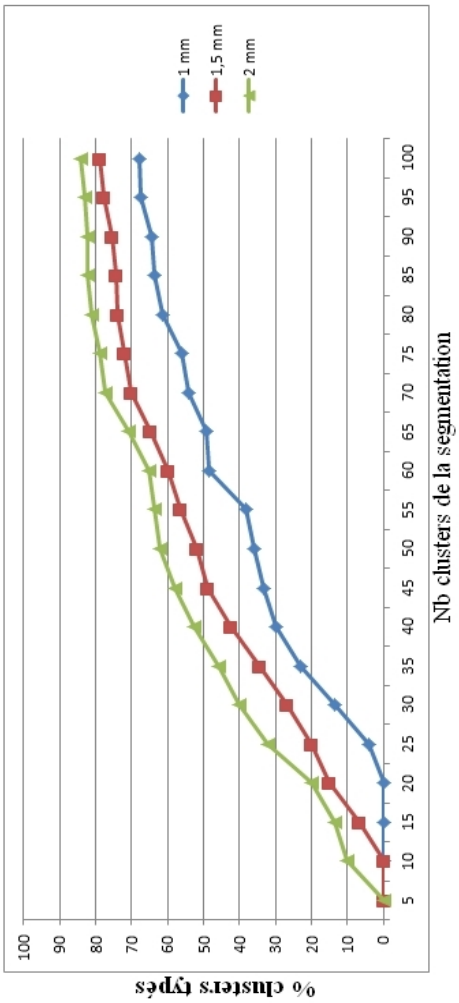


Tableau 4.3 – Évolution du nombre de surfaces ajustées par rapport au nombre de clusters pour un vilebrequin scanné.

Nous avons donc regardé en détails le nombre de surfaces ajustées à partir de 60 clusters en faisant le décompte du nombre de plans, de cylindres et de sphères ajustés par rapport au nombre de clusters à typer. Les résultats sont affichés dans le Tableau 4.4. Le graphique en-dessous du tableau représente le nombre de surfaces ajustées par type (plan, cylindre, sphère) avec en abscisse le nombre de clusters total du maillage. Nous remarquons de manière significative que le nombre de cylindres et de sphères stagne tandis que le nombre de plans continue de croître. Ces résultats concordent avec la méthode de segmentation employée dans Efpisoft. En effet, le nombre de clusters de type *plan* continue de croître jusqu'à atteindre le nombre de triangles du maillage (cf. description de la méthode à la section 3.2).

	Nb clusters	60	65	70	75	80	85	90	95	100
Bielle scannée (seuil 2mm)	Autre	90	91	93	93	94	95	96	96	96
	Plan	37	40	46	49	53	59	64	69	75
	Cylindre	2	3	3	5	6	6	6	6	6
	Sphère	15	16	16	16	16	16	16	16	15
Total surfaces ajustées		54	59	65	70	75	81	86	91	96
Pourcentage de surfaces ajustées		90	91	93	93	94	95	96	96	96

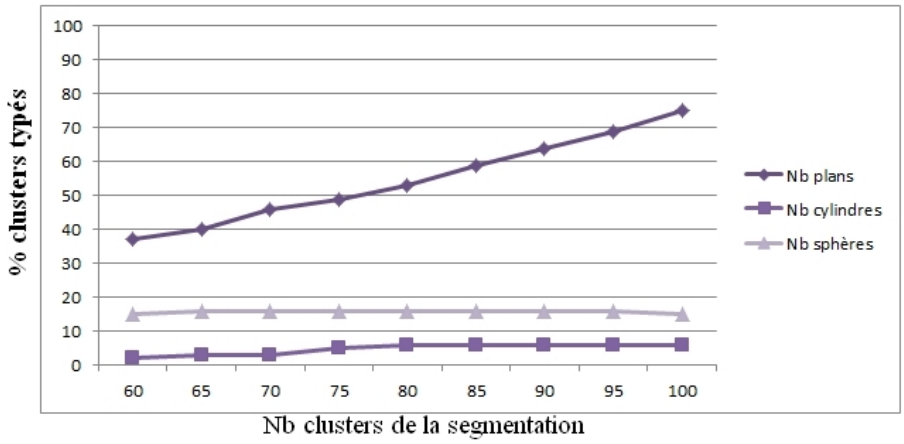


Tableau 4.4 – Évolution du nombre de surfaces ajustées de type plan, sphère et cylindre pour la bielle scannée.

Nous avons ensuite cherché une corrélation entre le nombre de clusters à partir duquel la courbe de pourcentage stagne (que nous avons nommé "limite") et le nombre de triangles total du maillage. Nous avons pour cela calculé le ratio suivant : *Nb clusters limite* ÷ *Nb triangles maillage* que nous avons ensuite ramené à un pourcentage. Nous retrouvons les résultats dans le tableau ci-dessous :

Composant	**Nb clusters limite**	**Nb triangles maillage**	**Ratio (en %)**
Bielle	60	41120	$0,15\%$
Piston	40	37437	$0,11\%$
Vilebrequin	80	67349	$0,12\%$

Cependant concernant les premiers tests réalisés, nous pouvons conclure les points suivants :

- lorsque le nombre de clusters est faible (inférieur à 15), l'étape de segmentation ne permet pas d'obtenir un niveau de d'information suffisant pour ajuster une surface sur chaque cluster. La signature de niveau 2 qui en résulte est un graphe contenant uniquement des nœuds de type *Autre* (non distincts les uns des autres).

- à partir d'une certaine valeur (propre à chaque maillage), l'augmentation du nombre de clusters n'améliore plus la segmentation. Les clusters tendent vers des plans et aucun autre type de surface n'est ajusté.

— la valeur du seuil ne peut encore être définie à ce stade. Son rôle ne sert qu'à "amplifier" les résultats. Une étude approfondie est à réaliser pour déterminer un seuil propre à chaque type de surface ajustée.

Notons que la valeur limite pour le vilebrequin est approximative. En effet, les tests auraient du être poursuivis au-delà de 100 clusters afin de valider cette valeur (limite). Les valeurs de ratio obtenues ne révèlent pas une corrélation significative (ratio compris entre $0,11$ et $0,15\%$) et il serait préférable d'appliquer ce test à un plus grand ensemble de composants afin d'en tirer des conclusions plus précises.

4.2 Expérimentation des mécanismes de signature

Dans cette section, nous présenterons les différents tests réalisés sur les solutions proposées pour l'étape de signature. Nous exposerons en particulier ceux réalisés sur la signature par graphe de niveau 2 ainsi que la signature par critère iso-périmétrique. En effet, ce sont les deux solutions qui ont été développées.

4.2.1 Pour la signature par graphe

Le but des tests sur la signature de niveau 2 est de déterminer la valeur du seuil de détection (évoqué dans la section précédente) et ceci pour chaque type de surface canonique (plan, cylindre ou sphère). Les tests présentés lors de l'expérimentation de la segmentation ont montré la nécessité de définir ces valeurs. Nous allons donc présenter la démarche expérimentale réalisée pour y parvenir.

Un jeu de données a été créé avec six maillages (issus de nuages de points) dont deux plans, deux sphères et deux cylindres. Pour chaque type, un maillage est issu d'une donnée CAO convertie en STL et l'autre d'un nuage de point issu du scan de pièces réelles. La figure 4.4 illustre les pièces réelles scannées.

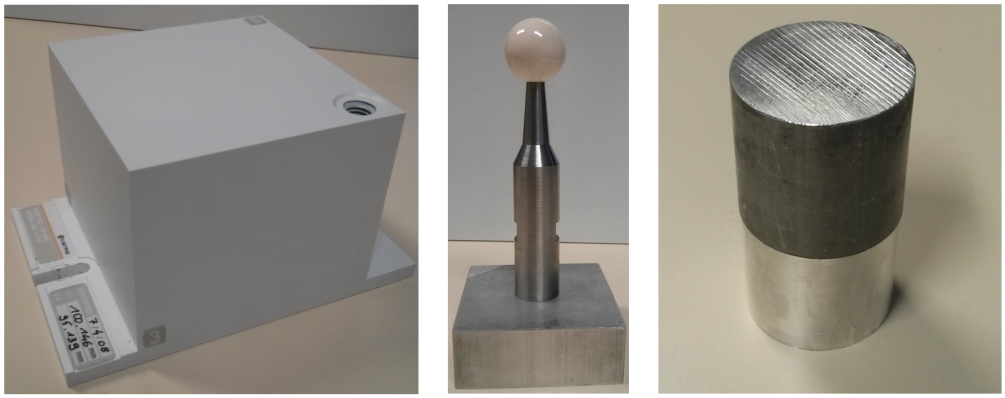


Figure 4.4 – Pièces réelles scannées pour les tests sur la signature par graphe de niveau 2.

La surface plane est scannée à partir d'un cube d'étalonnage pour bras laser (image de gauche de la figure 4.4). Au milieu, une sphère d'étalonnage de machine à mesurer tridimensionnelle a été utilisée. Puis à droite, une surface cylindrique en acier usinée a également servi. La figure 4.5 illustre les trois nuages de points correspondant aux pièces présentées.

L'acquisition des données a été réalisée à l'aide du bras laser de marque METRIS dans une salle climatisée à une température de 20,5°C. Pour rappel, la précision du bras laser est de ±0,040mm pour la mesure de longueur.

Nous avons ensuite extrait plusieurs caractéristiques dimensionnelles et géométriques de ces surfaces :

— longueur, largeur, diamètre et rayon ;

— nombre de points de chaque nuage.

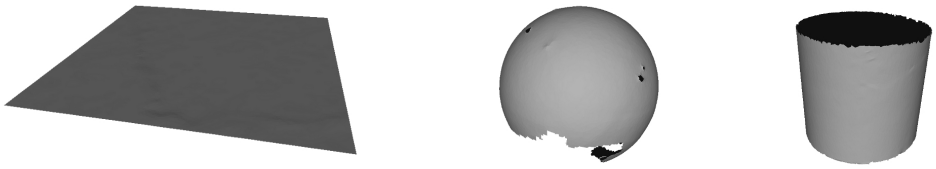


Figure 4.5 – Maillage issu du scan des pièces réelles présentées à la figure 4.4.

Grâce à ces informations, nous avons créé avec un modeleur CAO les surfaces correspondantes. Puis les fichiers ont été convertis en STL afin d'obtenir des maillages avec un nombre de points équivalent (à quelques centaines de points près). Pour réaliser cela, nous avons tout d'abord converti les surfaces CAO en STL avec un nuage de points très dense (nombre de points supérieurs à la donnée scannée). Les paramètres du logiciel (Catia V5-6 R2013) ont été modifiés pour notamment augmenter la résolution des pièces 3D. Une fois le fichier STL créé, nous l'avons importé dans RapidForm et nous avons utilisé l'outil "Decimate" afin de réduire le nombre de points jusqu'à la valeur souhaitée en faisant varier la densité du maillage.

Le tableau 4.5 récapitule l'ensemble des données utilisées ainsi que leurs caractéristiques respectives.

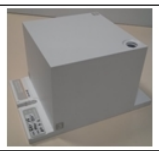
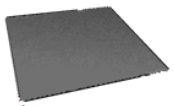
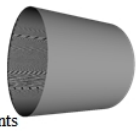
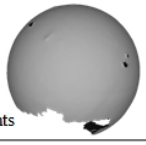
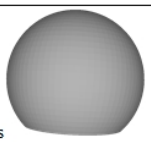
Donnée physique	**Donnée Scannée**	**Donnée CAO**	**Dimensions**
	 4780 points	 4820 points	60 x 60 mm
	 8329 points	 8319 points	Ø 34,99 mm hauteur 32,7 mm
	 6954 points	 7455 points	Ø29,9986 mm

Tableau 4.5 – Tableau récapitulatif des données utilisées pour les tests.

Hypothèse de départ Nous faisons l'hypothèse qu'une surface est ajustée lorsqu'au moins 90% de ses points appartiennent au modèle testé. En d'autres termes, le type (plan, sphère ou cylindre) est appliqué lorsque $pourcentage\_points > 90$, tel que défini à la section 3.3.1.2.

Principe du test Pour chaque donnée, les opérations suivantes ont été réalisées :

1. application de chaque algorithme (*FittingPlane* puis *FittingSphere* et *FittingCylinder*) ;
2. publication du pourcentage de points pour chaque algorithme ;
3. analyse des trois valeurs de pourcentage : on recherche la valeur la plus grande et on regarde si elle est strictement supérieure à 90. Si c'est le cas, alors le type correspondant (plan, sphère ou cylindre) est appliqué à la donnée. Si plusieurs valeurs sont strictement supérieures à 90, alors les types respectifs sont appliqués à la donnée. Si aucune valeur ne dépasse 90, alors le type *Autre* est attribué.

4. affichage du résultat grâce à une couleur primaire ou complémentaire : bleu pour un plan, vert pour une sphère et rouge pour un cylindre. Dans le cas où deux types sont attribués alors c'est la couleur complémentaire qui est appliquée. Par exemple, pour une donnée détectée comme plan et sphère en même temps, c'est la couleur cyan qui est utilisée. Enfin dans le cas où il y a les trois types (plan, sphère et cylindre), la couleur blanche est appliquée. Il en est de même pour le type *Autre*.

Variable du test La seule variable de ces tests est la valeur du seuil de détection de chaque type. Tel que décrit dans la section 3.3.1.2, il s'agit de calculer la distance euclidienne entre chaque point de la donnée et le modèle_possible. La valeur de cette distance est ensuite comparée à la valeur de ce seuil ($distance < seuil$). Pour chaque point, si l'inégalité est vérifiée, alors le point appartient au modèle (plan, sphère ou cylindre). Jusqu'ici la valeur de ce seuil n'a pu être définie, d'où le but de ce test. Plusieurs tests seront donc réalisés en faisant varier cette valeur entre 0 et 2 mm (0 / 0,02 / 0,05 / 0,1 / 0,2 / 0,5 / 1 / 1,5 / 2 mm).

Le tableau 4.6 présente les résultats obtenus. En colonne, nous avons le type de donnée testée (plan, sphère ou cylindre / scannée ou issue de la CAO) et sur les lignes, nous retrouvons les différentes valeurs de seuil. A chaque croisement ligne-colonne, on obtient deux résultats : tout d'abord le *Pourcentage_Points* obtenu pour chaque algorithme de *Fitting* (soit trois valeurs) puis en-dessous, nous retrouvons la couleur associée à l'analyse des trois valeurs obtenues avec le nom du(des) type(s) détecté(s) (cf. légende en dessous du tableau 4.6).

Nous allons maintenant décrire et analyser l'ensemble des résultats :

- pour les plans : seules les données scannées avec un seuil compris entre 0,05 et 0,2 sont associées à un plan. Les autres valeurs (0,5 / 1 / 1,5 / 2 mm) s'ajustent autant à un plan qu'à une sphère. Pour la donnée scannée, la valeur minimale possible pour le seuil est de 0,05mm, alors que la précision du bras laser est de ±0,040mm. Cette valeur-limite pour le seuil est donc cohérente. Concernant la donnée CAO, la valeur minimale pour le seuil est de 0,02mm. Ceci est dû à la donnée créée dans le modeleur CAO. En effet, pour créer chaque surface et la convertir en STL, il a fallu lui donner une épaisseur minimale de 0,01mm. Enfin concernant l'algorithme *FittingSphere* qui a détecté autant de sphères que de plans pour la donnée CAO (le pourcentage de points obtenu pour *FittingSphere* n'aurait pas dû être aussi élevé), il s'agit d'un problème géométrique que nous allons exposer.

En récupérant les valeurs numériques calculées dans l'algorithme, nous remarquons que pour un seuil de 0,5 mm, le rayon de la sphère idéale (modèle) est de $1,222 \times 10^3 mm$. Le plan est alors ajusté à un "morceau" d'une sphère de très grande dimension.

Intéressons-nous à la relation "rayon-corde-flèche" suivante :

$$f = r - \sqrt{r^2 - \frac{c^2}{4}} \tag{4.1}$$

avec f la longueur de la flèche, r le rayon de la sphère et c la longueur de la corde. La figure 4.6 illustre les composantes de l'équation 4.1. Le plan scanné, représenté par la corde, est ajusté à une sphère S de rayon r. La distance du plan jusqu'à la sphère est matérialisée par la flèche f. Ainsi la valeur de la distance entre chaque point de la donnée testée et la sphère idéale, doit être inférieure à la valeur de la flèche pour que le plan soit ajusté à celle-ci.

Donnée		Plan CAO			Plan SCAN			Sphère CAO			Sphère SCAN			Cylindre CAO			Cylindre SCAN	
Region																		
Dimensions (en mm)		Longueur	60		Longueur	60		Hauteur	32,7		Hauteur	32,7		Diamètre	29,999		Diamètre	29,999
		Largeur	60		Largeur	60		Diamètre	34,99		Diamètre	34,99						
Nb de points		4820			4780			8319			8329			7455			6954	
PourcentagePoints		Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre
Valeur seuil	2 mm	100	100	14.7305	100	100	15.0628	16.8903	100	63.4424	17.5919	100	62.3993	11.8283	56.2567	100	9.0811	40.1681	96.3249
type détecté		Plan - Sphère			Plan - Sphère			Sphère			Sphère			Cylindre			Cylindre		
Valeur seuil	1,5 mm	100	100	10.8982	100	100	10.8368	12.9461	100	57.8079	13.5833	99.8557	56.323	8.4505	36.5668	100	6.979	28.2522	82.1862
type détecté		Plan - Sphère			Plan - Sphère			Sphère			Sphère			Cylindre			Autre		
Valeur seuil	1 mm	100	100	7.5449	100	100	7.2385	9.2568	100	52.0392	9.3727	99.6250	48.6085	5.4334	24.6182	100	4.6006	18.02	62.2823
type détecté		Plan - Sphère			Plan - Sphère			Sphère			Sphère			Cylindre			Autre		
Valeur seuil	0,5 mm	100	100	3.7924	100	100	3.6611	4.9101	100	45.6533	4.7152	99.3222	37.3668	3.6663	10.7704	100	2.4144	8.9008	38.5345
type détecté		Plan - Sphère			Plan - Sphère			Sphère			Sphère			Cylindre			Autre		
Valeur seuil	0,2 mm	100	100	1.6766	100	74.8325	1.2343	2.2538	97.3436	16.1658	1.8169	98.6445	20.2019	1.5026	6.2507	100	0.97297	3.6035	19.2192
type détecté		Plan - Sphère			Plan			Sphère			Sphère			Cylindre			Autre		
Valeur seuil	0,1 mm	100	100	0.83832	100	38.353	0.50209	1.0196	95.4788	6.7346	0.90844	83.4318	10.3677	0.93761	4.6158	79.4687	0.49249	1.8978	11.1832
type détecté		Plan - Sphère			Plan			Sphère			Autre			Autre			Autre		
Valeur seuil	0,05 mm	100	100	0.35928	99.2463	19.3723	0.37657	0.45613	90.702	7.5664	0.54795	56.14	6.353	0.48083	4.6158	51.3413	0.28829	0.93693	6.1021
type détecté		Plan - Sphère			Plan			Sphère			Autre			Autre			Autre		
Valeur seuil	0,02 mm	100	100	0.11976	74.1841	7.9288	0.083682	0.1744	49.2755	4.6284	0.18745	19.2501	2.5667	0.20435	27.3711	27.3711	0.096096	0.42041	2.1021
type détecté		Plan - Sphère			Autre			Autre			Autre			Autre			Autre		
Valeur seuil	0 mm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
type détecté		Autre			Autre			Autre			Autre			Autre			Autre		

Tableau 4.6 – Résultats bruts obtenus pour la signature de niveau 2 sur le jeu de données.

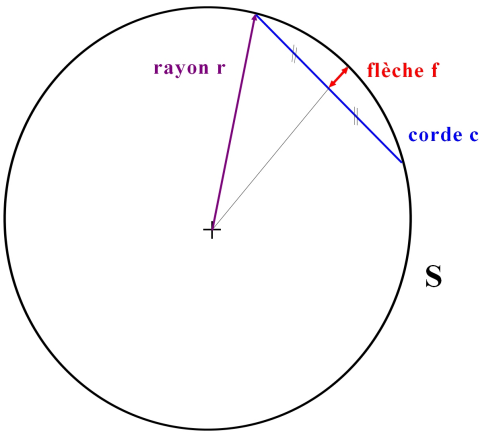


FIGURE 4.6 – Relation rayon-corde-flèche pour une sphère ajustée à une surface plane.

En prenant, $r = 1,222 \times 10^3 mm$ et $c = 60mm$ (la largeur du plan), nous obtenons :

$$f = 1,222 \times 10^3 - \sqrt{(1,222 \times 10^3)^2 - \frac{60^2}{4}}$$

$$f = 0,368304mm$$

La valeur obtenue pour f indique que pour $seuil > 0,36mm$, la donnée testée est détectée comme une sphère de très grand rayon. D'après le tableau 4.6, le score obtenu pour *FittingSphere* pour un seuil de 0,5mm est de 100% puis de 74,8% pour un seuil 0,2mm, ce qui confirme la valeur de la flèche obtenue précédemment. Remarquons qu'en-dessous de cette valeur de seuil, le plan scanné n'est plus détecté comme sphère. Cela découle du fait que la surface plane scannée est légèrement irrégulière.

- **pour les sphères :** la sphère scannée est détectée à partir d'un seuil $\geq 0,2mm$ alors que pour la sphère de référence (CAO), il est possible d'en détecter une à partir de 0,05mm. Nous pensons que cette différence est due à la qualité du nuage de points scanné.

- **pour les cylindres :** seul le seuil de 2 mm a permis d'ajuster un cylindre sur la donnée scannée. Pour ce qui est du cylindre de référence (issu de la CAO), il est possible d'ajuster ce type de surface jusqu'à un seuil de 0,2 mm. Cette valeur étant plus grande que toutes les autres, nous avons donc analysé l'algorithme FittingCylinder pas à pas avec le cylindre de référence. Pour cela, nous avons réalisé trois tests avec les valeurs de seuil suivantes : 1,5 mm, 0,1 mm et 0,02 mm. La figure 4.7 présente l'évolution des données au fur et à mesure de l'algorithme. Sur la ligne A, nous retrouvons le cylindre de référence avec en bleu les points appartenant au plan qui le coupent latéralement. Nous remarquons que pour les seuils 0,1 et 0,02 mm, le nombre de points bleus est restreint. En effet, la sélection de ces points est réalisée en calculant la distance des points du nuage par rapport au plan de coupe. Cette distance est comparée à la valeur du seuil. Plus elle est petite et moins de points sont proches du plan. Ceci découle du fait que l'équation du plan est calculée mathématiquement et que ce dernier ne passe pas exactement par les points du nuage.

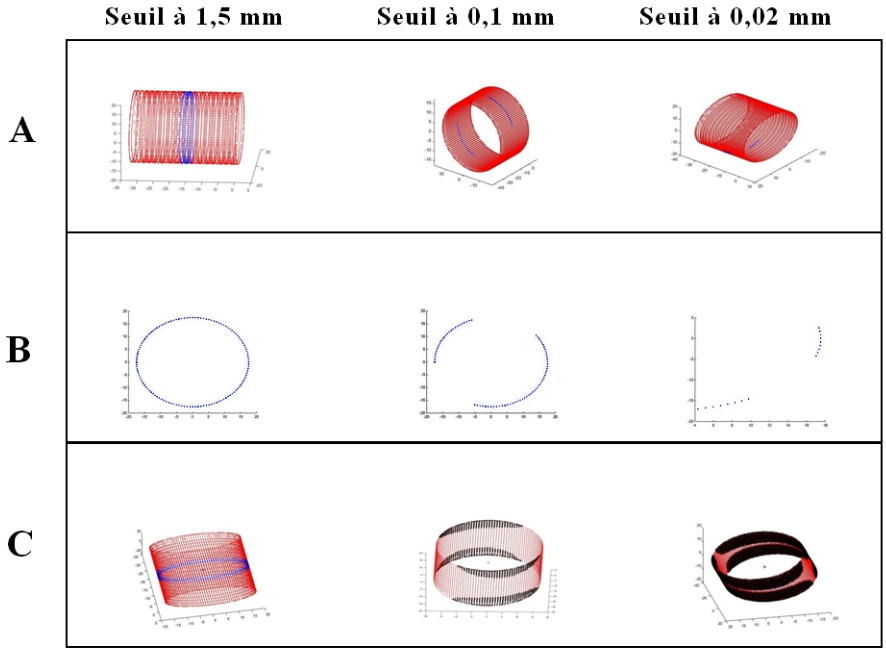


FIGURE 4.7 – Test du cylindre de référence avec l'algorithme FittingCylinder.

Sur la ligne B, nous avons les points sélectionnés et projetés sur le plan. Nous remarquons que pour un seuil de 0,02 mm, l'arc dessiné par la projection des points possède une forme légèrement elliptique. Ceci provoque un déplacement léger du centre du "cercle" et par conséquent du cylindre. Enfin sur la ligne C, les points en noir représentent les points de la donnée dont la distance par rapport au modèle est supérieure au seuil. De la même manière qu'avec le plan, le cylindre de référence a été créé dans le modeleur CAO avec une épaisseur de 0,01 mm. En fonction de la couche intérieure ou extérieure sur laquelle se situe chaque point, ils appartiennent ou non au modèle. Plus le seuil est faible et plus le nombre de points appartenant au modèle diminue. De plus, l'erreur commise par le déplacement du centre (arc de cercle elliptique) accentue le phénomène. En effet, c'est le rayon de ce cercle qui sert de référence pour comparer la distance entre chaque point de la donnée et l'axe du cylindre.

Suite aux différentes remarques et problèmes soulevés dans l'analyse des résultats, nous proposons les améliorations suivantes :

- pour l'algorithme FittingSphere : une condition (filtre) est ajoutée pour éviter que les plans soient ajustés à des sphères de grand rayon. Les fonctions et opérations suivantes ont été insérées dans l'algorithme :

Algorithme 3 Améliorations apportées à FittingSphere

Calcul de la boîte englobante orientée de la donnée (cluster ou région)
Calcul de la diagonale de la boîte
Calcul du ratio $= \dfrac{RayonSphere}{longueur\_diagonale\_boite}$
if $ratio < 5$ **then**
  Calcul de $Pourcentage\_Points$
else
  $Pourcentage\_Points = 0$
end if

Le calcul de ce ratio permet d'écarter le calcul de FittingSphere et ainsi éviter que des plans soient ajustés à des sphères. Le choix de la valeur maximale de ce ratio (5) a été défini de manière empirique.

- pour l'algorithme FittingCylinder : d'autres tests sur la donnée scannée seront réalisés avec de nouvelles valeurs de seuil afin d'ajuster plus précisément (au 10ème de millimètre) cette valeur.

Après application de l'algorithme 3 au code initial, nous avons ré-effectué les tests et les résultats sont présentés dans le tableau 4.8. Notons que tous les plans sont colorés en bleu foncé (le type "sphère" ne s'ajustant plus). Les autres résultats ne sont pas affectés.

Pour ce qui concerne les surfaces cylindriques scannées, les nouveaux tests ont été réalisés avec les valeurs de seuil suivantes : 2 / 1,9 / 1,8 / 1,7 / 1,6 / 1,5 mm. Nous pouvons observer dans le tableau de la figure 4.9 que le cylindre scanné est détecté jusqu'à un seuil de 1,8 mm.

L'expérimentation des algorithmes sur cet ensemble de données a permis de fixer des valeurs de seuil pour chaque type de *Fitting*. D'après les résultats obtenus dans le tableau 4.8 et la figure 4.9, nous gardons les valeurs minimales de seuil pour lesquelles une surface est ajustée avec au moins 90% de points appartenant au modèle et ceci pour les données scannées soit :

— pour les plans : $seuil_1 = 0,05mm$
— pour les sphères : $seuil_2 = 0,2mm$
— pour les cylindres : $seuil_3 = 1,8mm$

Ces trois valeurs sont ensuite insérées dans l'algorithme principal de signature de niveau 2. Pour finir, nous avons testé à nouveau la bielle scannée présentée dans la section 4.1. Nous avons réalisé le même test dont les résultats sont présentés dans le tableau 4.1.

	Nb clusters	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Bielle scannée	Autre	5	8	13	18	23	28	33	37	42	46	50	53	58	62	63	66	69	74	76	79
	Plan	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Cylindre	0	2	2	2	2	2	2	3	3	4	5	7	7	8	12	14	16	16	19	21
	Sphère	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total surfaces ajustées		0	2	2	2	2	2	2	3	3	4	5	7	7	8	12	14	16	16	19	21
Pourcentage de surfaces ajustées		0	20	13,3	10	8	6,67	5,71	7,5	6,67	8	9,09	11,7	10,8	11,4	16	17,5	18,8	17,8	20	21

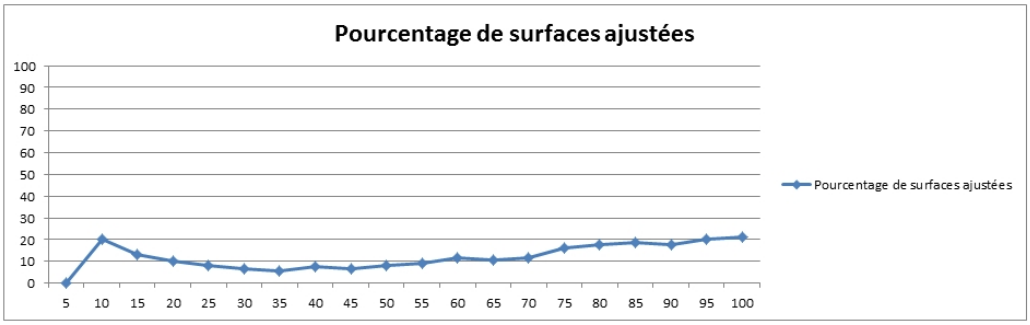


Tableau 4.7 – Évolution du nombre de surfaces ajustées par rapport au nombre de clusters pour une bielle scannée après amélioration des algorithmes.

En comparant les valeurs obtenues pour la bielle scannée dans le tableau 4.7 avec celles obtenues dans la section 4.1, nous remarquons que la courbe de l'évolution du nombre de surfaces ajustées présente une pente beaucoup plus faible que précédemment. Précédemment avec 60 clusters, 90% des surfaces étaient ajustées, désormais seulement 11,7% le sont. De nouveaux tests seraient nécessaires pour déterminer le nouveau nombre de clusters à partir duquel le nombre de surfaces ajustées stagne. Cependant ces tests feront partie des perspectives.

Donnée	Plan CAO			Plan SCAN			Sphère CAO			Sphère SCAN			Cylindre CAO			Cylindre SCAN	
Region																	
Dimensions (en mm)	Longueur		60	Longueur		60	Hauteur		32,7	Hauteur		32,7	Diamètre		29,9986	Diamètre	29,9986
	Largeur		60	Largeur		60	Diamètre		34,99	Diamètre		34,99					
Nb de points	4820			4780			8319			8329			7455			6954	
PourcentagePoints	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre	Plan	Sphère	Cylindre
Valeur seuil	100	0	14.7305	100	0	15.0628	16.8903	100	63.4424	17.5919	100	62.9993	11.8283	56.2567	100	9.0811	40.1681	96.9249
2 mm	Plan			Plan			Sphère			Sphère			Cylindre			Cylindre		
type détecté																		
Valeur seuil	100	0	10.8982	100	0	10.8368	12.9461	100	57.8079	13.5833	99.8557	56.323	8.4505	36.5668	100	6.979	28.2522	82.1862
1,5 mm	Plan			Plan			Sphère			Sphère			Cylindre			Autre		
type détecté																		
Valeur seuil	100	0	7.5449	100	0	7.2385	9.2568	100	52.0392	9.3727	99.6250	48.6085	5.4934	24.6182	100	4.6006	18.02	62.2823
1 mm	Plan			Plan			Sphère			Sphère			Cylindre			Autre		
type détecté																		
Valeur seuil	100	0	3.7924	100	0	3.6611	4.9101	100	45.6533	4.7152	99.3222	37.9668	3.6663	10.7704	100	2.4144	8.9008	38.5345
0,5 mm	Plan			Plan			Sphère			Sphère			Cylindre			Autre		
type détecté																		
Valeur seuil	100	0	1.6766	100	0	1.2343	2.2538	97.3436	16.1658	1.8169	98.6445	20.2019	1.5026	6.2507	100	0.97297	3.6035	19.2192
0,2 mm	Plan			Plan			Sphère			Sphère			Cylindre			Autre		
type détecté																		
Valeur seuil	100	0	0.83832	100	0	0.50209	1.0196	95.4788	6.7346	0.90844	83.4318	10.3677	0.93761	4.6158	79.4687	0.49249	1.8978	11.1832
0,1 mm	Plan			Plan			Sphère			Autre			Autre			Autre		
type détecté																		
Valeur seuil	100	0	0.35928	99.2469	0	0.37657	0.45613	90.702	7.5664	0.54795	56.14	6.359	0.48083	4.6158	51.9413	0.28829	0.93693	6.1021
0,05 mm	Plan			Plan			Sphère			Autre			Autre			Autre		
type détecté																		
Valeur seuil	100	0	0.11976	74.1841	0	0.083682	0.1744	49.2755	4.6284	0.18745	19.2501	2.5667	0.20435	27.3711	27.3711	0.096096	0.42041	2.1021
0,02 mm	Plan			Autre			Autre			Autre			Autre			Autre		
type détecté																		
Valeur seuil	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 mm	Autre			Autre			Autre			Autre			Autre			Autre		
type détecté																		

Figure 4.8 – Résultats de la signature de niveau 2 après amélioration des algorithmes.

		Cylindre SCAN		
				
		Hauteur		32,7
		Diamètre		34,99
		8329		
		Plan	Sphère	Cylindre
Valeur seuil	**2 mm**			96.9250
	type détecté	Cylindre		
Valeur seuil	**1,9 mm**			94.2343
	type détecté	Cylindre		
Valeur seuil	**1,8 mm**			92.6127
	type détecté	Cylindre		
Valeur seuil	**1,7 mm**			89.1292
	type détecté	Autre		
Valeur seuil	**1,6 mm**			85.7058
	type détecté	Autre		
Valeur seuil	**1,5 mm**			82.1863
	type détecté	Autre		

FIGURE 4.9 – Expérimentation de l'algorithme FittingCylinder sur le cylindre scanné avec de nouvelles valeurs de seuil.

Pour conclure, ces tests ont permis d'expérimenter notre algorithme de signature de niveau 2 et de l'améliorer. Cependant les résultats obtenus laissent supposer que cette amélioration risque d'augmenter fortement le nombre de clusters et par conséquent le nombre de nœuds des graphes afin d'obtenir un nombre suffisant de surfaces ajustées (informations qui caractérisent la signature de niveau 2). Comme nous le verrons dans la section 4.3, le nombre de clusters a un fort impact sur la comparaison des signatures et en particulier, la comparaison des graphes. Plus ces derniers possèdent un nombre important de nœuds, plus il devient impossible de les comparer (temps de calcul important, dépassement des capacités de calcul des machines).

4.2.2 Pour le critère iso-périmétrique

Hypothèse Nous faisons l'hypothèse que les données utilisées dans les tests pour le critère iso-périmétrique sont équivalentes à des données acquises par scan. En effet, au vue de la quantité de données nécessaires pour réaliser ces tests, nous avons utilisé des données issues de modeleurs CAO que nous avons ensuite converties au format STL.

Dans cette section, nous présenterons les différents tests qui découlent de l'application de la formule 3.1 du critère iso-périmétrique. D'un test à l'autre, les variables ainsi que les données testées sont différentes. La description de chaque test est la suivante :

Test 1 : sur plusieurs familles de composants comme par exemple : les bielles, les vilebrequins, les pistons, etc. (voir figure 4.10). Les résultat sont présentés dans le tableau 4.8 ;

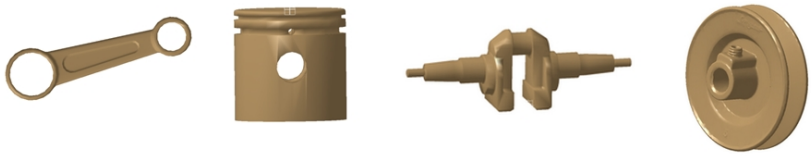


Figure 4.10 – Exemple de familles de composants testés pour la signature par critère iso-périmétrique.

Test 2 : sur une partie des composants présentés précédemment mais en faisant varier la complétude de chaque donnée. Tout d'abord les surfaces fonctionnelles ont été enlevées sur chaque composant ce qui correspond aux surfaces manquantes dans un assemblage scanné. Puis les nuages ont été tronqués (réduction du nombre initial de points) pour simuler les assemblages scannés "sans démontage" où des composants en cachent d'autres. Les résultats sont présentés dans les tableaux 4.9, 4.10 et 4.11 ;

Test 3 : sur des assemblages mécaniques de différents types. Les résultats sont présentés dans le tableau 4.12 ;

Test 4 : sur un des assemblages mécaniques précédent mais en faisant varier la position des pièces les unes par rapport aux autres. C'est pour simuler le déplacement possible des pièces dans un assemblage. Les résultats sont dans le tableau 4.13.

Pour le **test 1**, nous remarquons que suivant la famille, les valeurs peuvent être regroupées par intervalle de valeurs. En présentant sur les résultats de trois familles sur une échelle graduée et en affectant un marqueur différent à chaque famille, nous obtenons le graphe de la figure 4.11.

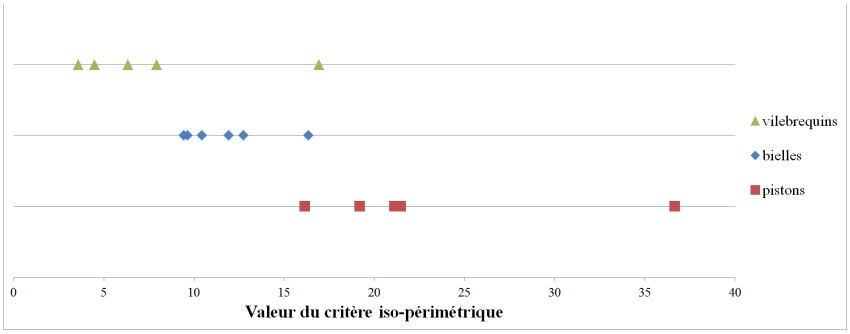
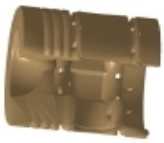


Figure 4.11 – Répartition des valeurs de critère iso-périmétrique pour trois familles de composants : bielle, piston et vilebrequin.

En écartant les valeurs isolées, nous obtenons les intervalles suivants :
— pour les vilebrequins : de 3,58 à 7,91
— pour les bielles : de 9,63 à 12,73
— pour les pistons : de 16,16 à 21,47.

Nom de la pièce	bielle-1.stl	bielle-2.stl	bielle-3.stl	bielle-4.stl	bielle-5.stl	Bielle-6.stl
Image						
Valeur C(D)	12,73	10,44	16,34	9,63	11,91	9,44

Nom de la pièce	Piston-1.stl	Piston-2.stl	Piston-3.stl	Piston-4.stl	Piston-5.stl	Piston-6.stl
Image						
Valeur C(D)	19,19	19,19	21,1243	21,47	36,66	16,16

Nom de la pièce	vilebrequin-1.stl	vilebrequin-2.stl	vilebrequin-3.stl	vilebrequin-4.stl	vilebrequin-5.stl	poulie_tracepart1.stl
Image						
Valeur C(D)	6,33	4,48	3,38	16,92	7,91	33,11

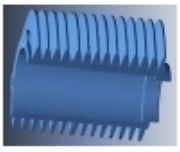
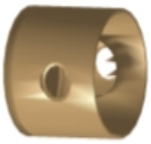
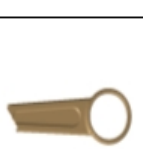
Nom de la pièce	poulie_tracepart2.stl	poulie_tracepart3.stl	poulie_ifpen.stl	alternateur_ifpen.stl	carter_1	carter_2
Image						
Valeur C(D)	13,33	15,35	9,48	6,17	12,28	56,45

Tableau 4.8 – Application du critère iso-périmétrique à plusieurs familles de composants.

Nom de la pièce	bielle-1.stl	bielle-2.stl	bielle-3.stl	Piston-1.stl	Piston-2.stl	Piston-3.stl
Image (composant sans surface fonctionnelle)						
Valeur C(D)	7,1	6,84	8,09	18,87	16,72	17,45

Nom de la pièce	vilebrequin-1.stl	vilebrequin-2.stl	vilebrequin-3.stl
Image (composant sans surface fonctionnelle)			
Valeur C(D)	6,4	4,8	3,74

Tableau 4.9 – Application du critère iso-périmétrique en enlevant les surfaces fonctionnelles.

Nom de la pièce	bielle-1.stl	bielle-2.stl	bielle-3.stl	Piston-1.stl	Piston-2.stl	Piston-3.stl
Image (composant complet à 70%)						
Valeur C(D)	9,72	9,96	12,55	9,49	11,17	11

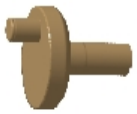
Nom de la pièce	vilebrequin-1.stl	vilebrequin-2.stl	vilebrequin-3.stl	poulie_tracepart1.stl
Image (composant complet à 70%)				
Valeur C(D)	6,14	3,21	4,56	26,75

Tableau 4.10 – Application du critère iso-périmétrique avec des données complètes à 70%.

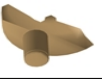
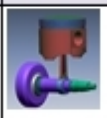
Nom de la pièce	bielle-1.stl	bielle-2.stl	bielle-3.stl	Piston-1.stl	Piston-2.stl	Piston-3.stl	vilebrequin-1.stl	vilebrequin-2.stl	vilebrequin-3.stl	poulie_tracepart1.stl
Image (composant complet à 30%)										
Valeur C(D)	8,65	11,18	1,51	11,13	11,25	4,98	2,97	4,01	1,77	20,82
Image (composant complet à 50%)										
Valeur C(D)	9,23	3,03	7,85	2,55	4,23	3,11	2,89	4,01	4,09	17,88
Image (composant complet à 10%)										
Valeur C(D)	3,15	5,15	2,78	3,23	3,25	3,56	2,71	1,56	2,14	3,94

Tableau 4.11 – Application du critère iso-périmétrique avec des données complètes à 30, 50 et 10%.

Nom de l'assemblage	Assemblage-1	Assemblage-2	Assemblage-3	Assemblage-4	Bielle-piston-IFPEN	demi_train_avant	courroie-poulie
Image							
Valeur C(D)	17,09	14,34	17,87	9,85	23,83	20,46	19,19

Nom de l'assemblage	Assemblage_carburateur	assemblage_divers-1	assemblage_divers-2	assemblage_freinvtt	assemblage_divers-3	moteur_recallé IFPEN	moteur_recallé complet
Image							
Valeur C(D)	16,88	7,25	12	6,6	4,51	16,69	15,91

Nom de l'assemblage	horloge_tn22	ventilo_1	ventilo_3	serre_joint_tn20	cle_01	moteur_tn20	projet_buggy_tn20
Image							
Valeur C(D)	11,93	4,87	7,01	11,72	14,38	19,26	21,21

Nom de l'assemblage	Frein_a_generatrice	embrayage_electromagnetique	antideriveur à came	moteur_1	cle_02	moteur_2	moteur_3
Image							
Valeur C(D)	20,98	15,09	52,77	35,42	17,46	36,88	19,06

Nom de l'assemblage	pompe-1	pompe-2	pompe-3	pompe-4	pompe-5	reducteur-1	pompe_electrique
Image							
Valeur C(D)	19,1	18,21	20,34	9,39	20,53	12,28	29,72

Nom de l'assemblage	motoréducteur-1	motoréducteur-2	reducteur-2	reducteur-3	reducteur-4	reducteur-5	reducteur_6
Image							
Valeur C(D)	12,02	8,82	8,31	19,39	12,17	7,87	27,19

Nom de l'assemblage	reducteur-7	reducteur-8	reducteur-9	boite_de_vitesse	rabotteuse_bois	servomoteur
Image						
Valeur C(D)	12,11	6,54	58,68	16,09	26,4	23,99

Tableau 4.12 – Application du critère iso-périmétrique avec des assemblages mécaniques.

Nom de l'assemblage	Ensemble.Attelage.Mobile-1	Ensemble.Attelage.Mobile-2	Ensemble.Attelage.Mobile-3
Image			
Valeur C(D)	6,05	5,87	7,57

Nom de l'assemblage	Ensemble.Attelage.Mobile-4	Ensemble.Attelage.Mobile-5
Image		
Valeur C(D)	7,37	10,46

Tableau 4.13 – Application du critère iso-périmétrique en faisant varier la position des pièces dans un assemblage.

Le **test 2** consiste à faire varier la complétude des données. Nous avons d'abord comparé les données entières avec les données sans surface fonctionnelle. D'après le tableau 4.9, nous notons que les valeurs obtenues pour les pistons et les vilebrequins testés entrent dans les intervalles de valeurs relatifs au test 1. Seules les valeurs correspondantes aux bielles ne coïncident pas. La figure 4.12 illustre la comparaison entre les résultats du tableau 4.8 (données à 100%) et les données nommées ci-dessus. Seules les données du tableau 4.8 correspondant aux données testées au test 2 ont été ajoutées au graphique de la figure 4.12.

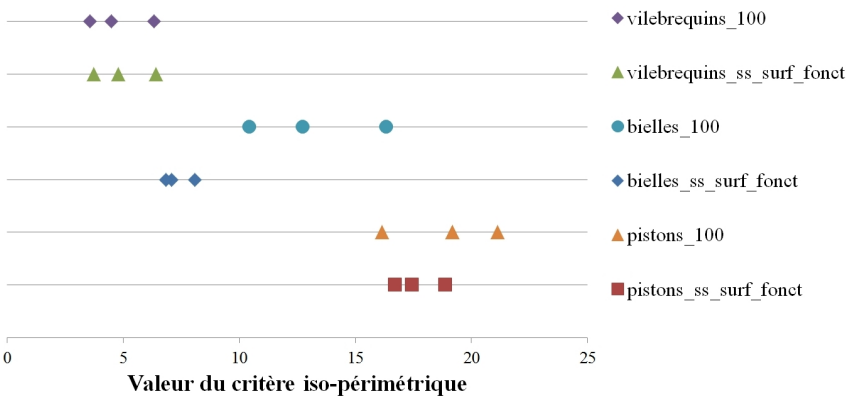


FIGURE 4.12 – Répartition des valeurs du critère iso-périmétrique pour des composants sans surface fonctionnelle (test 2) par rapport aux valeurs obtenues pour les données à 100% correspondantes.

Ensuite, nous avons fait varier le nombre de points de chaque donnée de 70 à 10%. Nous pouvons comparer les différentes valeurs grâce à la figure 4.13 où les valeurs de références (données à 100%) ont été ajoutées à la comparaison. Nous nous apercevons que les valeurs obtenues sont dispersées pour les bielles et les pistons. Seules les valeurs obtenues pour les vilebrequins restent "groupées" sur chaque ligne. Cependant ces valeurs n'entrent plus dans les intervalles définis précédemment pour chaque famille de composant.

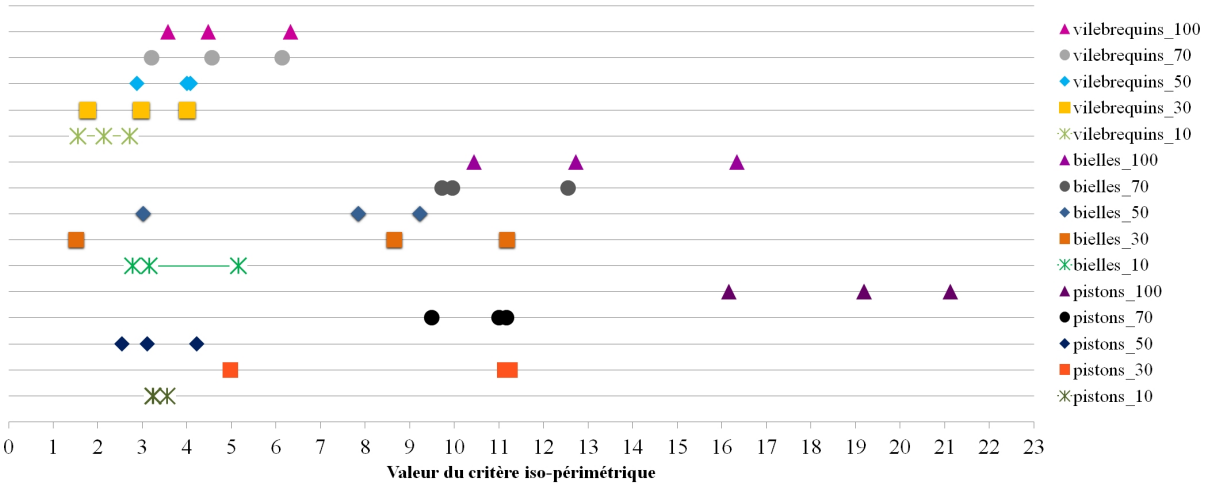


FIGURE 4.13 – Évolution des valeurs du critère iso-périmétrique par rapport à la complétude des données.

Si nous comparons les valeurs avec la même complétude (10% par exemple), nous remarquons que les valeurs du critère iso-périmétrique (toutes familles de composants confondues) tendent vers 0 et il devient impossible de distinguer un composant d'un autre.

Pour le **test 3**, la formule du critère iso-périmétrique a été appliquée à des assemblages mécaniques : pompes, moteurs, réducteurs etc. Pour comparer les résultats, nous nous sommes inspirés du formalisme des graphes de [Ashby, 2005] qui permettent de classer des matériaux en fonction de leurs propriétés mécaniques. Pour cela, nous avons classé les assemblages par catégories et nous les avons mis en abscisse. Puis en ordonnée, nous avons créé une échelle graduée pour les valeurs du critère iso-périmétrique. Au croisement de chaque ligne-colonne, nous avons mis la valeur obtenue pour chaque assemblage selon sa catégorie. Enfin, nous avons pris les valeurs extremums de chaque catégorie pour créer un cadre coloré représentant l'intervalle de valeurs pour la catégorie associée. La figure 4.14 illustre cela.

Nous pouvons remarquer que toutes les catégories se chevauchent et qu'il n'est pas possible de les distinguer les unes des autres comme cela fut possible pour le test 1. En d'autres termes, si nous testons une nouvelle donnée et que la valeur de son critère iso-périmétrique est comprise entre 12 et 21, nous ne pouvons pas déterminer à quelle catégorie il pourrait appartenir. Au mieux, il serait associé à trois catégories (sur les 9).

Lorsque nous nous intéressons de plus près aux valeurs obtenues pour des données scannées (avec * et **), nous observons que ces valeurs ont tendance à allonger les intervalles de valeurs pour leur catégorie respective. Nous pouvons nous interroger sur le fait de mélanger les données issues de CAO avec celles obtenues à partir de vrai scan.

Pour le dernier test (**test 4**), nous avons repris un des assemblages (ensemble attelage mobile) et nous avons fait varier d'abord la position de la bielle par rapport au vilebrequin puis la bielle et le piston par rapport au troisième composant. Nous pouvons remarquer que le changement de position des pièces influe peu sur la valeur du critère iso-périmétrique. Nous obtenons une différence de 0,27 entre *Ensemble.Attelage.Mobile-1* et *Ensemble.Attelage.Mobile-2* (rotation de 45 degrés de la bielle par rapport au vilebrequin) et une différence maximale de 2,89 entre *Ensemble.Attelage.Mobile-3*, *Ensemble.Attelage.Mobile-4* et *Ensemble.Attelage.Mobile-5* ; ceci malgré que la bielle et le piston aient tourné d'environ 140 degrés autour du vilebrequin.

Pour conclure, ces quatre tests ont permis d'expérimenter notre signature par critère global dans différentes situations : en faisant varier la complétude des données et la position des composants dans un assemblage. Le but est de simuler des situations réelles afin de tester la robustesse de notre solution. Les résultats obtenus ne permettent pas de conclure positivement sur l'utilisation de cette signature pour identifier des assemblages mécaniques. Il serait préférable de continuer à enrichir notre graphe de type Ashby avec de nouveaux assemblages ou notre tableau 4.8 avec de nouveaux pistons, vilebrequins et bielles pour valider les intervalles de valeurs que nous avons utilisés comme référence.

De plus, l'écart des valeurs entre composants scannés et composants issus de la CAO nous amène à croire qu'il serait préférable de re-créer un graphe de Ashby mais uniquement avec des valeurs issues de scans.

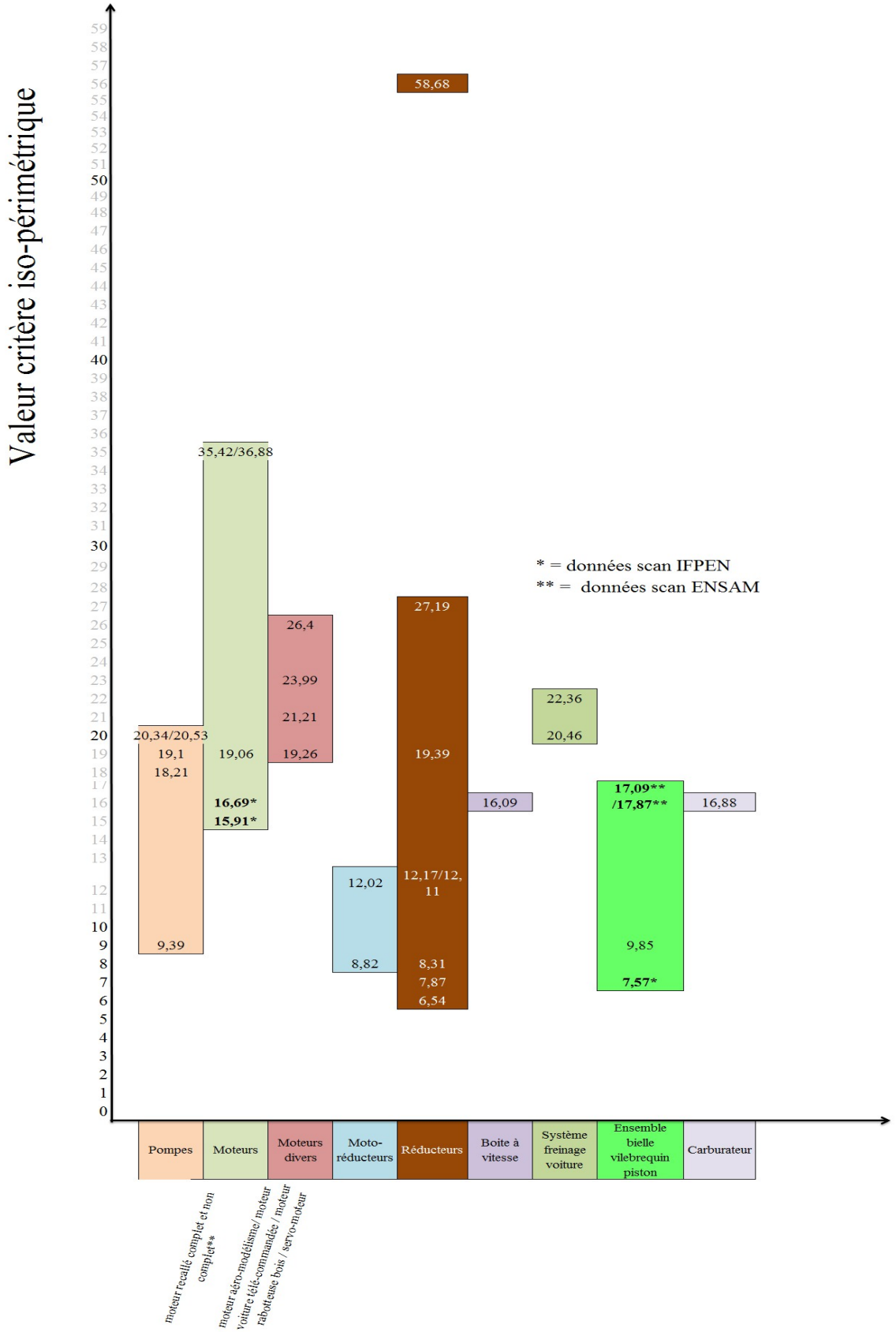


FIGURE 4.14 – Comparaison des valeurs obtenues pour le test 3 grâce à un graphe inspiré de [Ashby, 2005].

4.3 Expérimentation de la comparaison

Dans cette section, nous testerons nos algorithmes de comparaison de signature et particulier ceux de comparaison de graphes (signature de niveaux 1 et 2). L'ensemble des expérimentations n'ayant pas été réalisé dans le même ordre que celui du manuscrit, les signatures utilisées pour ces tests ont été générées avec un seuil de détection de 1,5 mm (cf. section 4.2.1).
Nos tests sont en deux parties :

— **1ère phase de tests :** nous avons testé deux composants scannés et nous avons comparé leurs signatures (niveaux 1 et 2) à celles des composants correspondants issus de la base de connaissances. Ces dernières ont été générées à partir de modèles CAO puis convertis au format STL. La figure 4.15 illustre les couples de données testées. Le but de ce test est d'examiner la similarité entre les deux signatures et d'évaluer la robustesse de l'algorithme pour le premier niveau de signature (section 4.3.1) puis pour le deuxième (section 4.3.2).

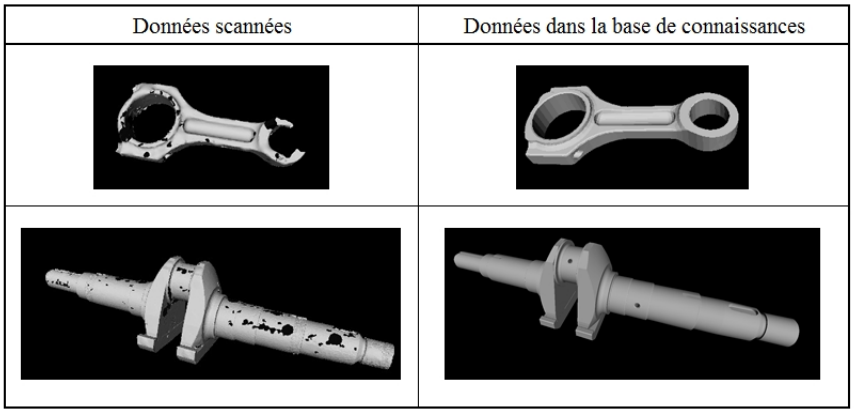


FIGURE 4.15 – Maillages utilisés pour la première phase de tests de comparaison.

— **2ème phase de tests :** cette deuxième série de tests a consisté à expérimenter les capacités de calcul entre un assemblage de données scannées et un ensemble de données issues de la base de connaissances. Le but de ce test est de donner un ordre de grandeur du coût de comparaison (temps nécessaire) entre deux signatures, en fonction notamment du nombre de nœuds du graphe et ceci pour les deux niveaux de signature (section 4.3.3).

Remarque : Tout au long des tests et des résultats présentés dans cette section, nous attacherons de l'importance au nombre de clusters typés [1] dans les *mapping* issus de chaque comparaison. En effet, un *mapping* avec un nombre de clusters non-typés important posera des problèmes pour la comparaison de niveau 3. De plus, le fait de comparer des signatures avec un niveau d'information bas (avec peu de clusters typés), nous pourrions imaginer avoir des résultats faux-vrais (composant identifié par erreur).

Hypothèse 1 : Nous ferons l'hypothèse qu'une signature de niveau 2 est viable dès lors qu'au moins un des nœuds de son graphe est typé. De notre point de vue, nous considèrerons que plus le nombre de nœuds typés est important, plus la signature sera de qualité.

Hypothèse 2 : Une comparaison sera considérée comme viable si au moins, l'un de ses *mapping* (liste de correspondances) contient au moins un nœud typé.

1. Cluster typé : cluster qui a été défini comme plan, cylindre ou sphère dans l'étape de signature de niveau 2. Dans le cas d'un graphe, il s'agit de l'attribut d'un nœud (cf. section 3.3.1).

Préparation des données : Afin de réaliser ces tests, les opérations suivantes ont été réalisées sur les quatre maillages de la figure 4.15 afin de créer les différents jeux de données :

- segmentation de la donnée scannée en un nombre de clusters donné : 5, 10, 15, 20 etc. jusqu'à 100 clusters. Chaque configuration (ex : 15 clusters) équivaut à une donnée.
- signature de chaque donnée segmentée pour le niveau 1 puis le niveau 2.

En fonction du test, seule une partie des données a été utilisée.

Résultats espérés / idéal : Dans notre idéal, nous imaginons par exemple qu'un composant scanné de 20 clusters "matche" avec le composant équivalent dans la base de connaissances (nombre de clusters +/- égal) et ceci avec un score de similarité n°2 (cf. $S_{G'_1}$ dans la section 3.4) qui soit maximal par rapport aux autres (environ 80%).

4.3.1 Comparaison de niveau 1 pour la bielle et le vilebrequin

Nous avons d'abord commencé par réaliser les tests avec la bielle.

Jeux de données :

- 1er jeu de données : pour la bielle scannée, nous avons un nombre de clusters compris entre 6 et 10 (6, 7, 8, 9 et 10 clusters). Le nombre de clusters est limité à 10 du fait des limites de calcul de la machine utilisée [2] pour ces tests. Les résultats présentés sur les pages suivantes correspondent donc aux tests qui ont pu être réalisés. La section 4.3.3 reprendra en détails les tests sur les limites de calcul.
- 2eme jeu de données : pour la bielle issue de la CAO, nous avons utilisé 14 signatures comprises entre 5 et 100 clusters.

Principe du test : Nous avons réalisé les opérations suivantes :

- pour chaque donnée de X clusters du premier jeu, nous l'avons comparé à l'ensemble des données du deuxième jeu de données. Une minuterie a été configurée pour stopper le calcul au bout de 5 heures.
  NB : initialement il n'y avait aucune minuterie dans l'algorithme, cependant la comparaison d'une seule signature pouvait durer jusqu'à plusieurs jours voire une semaine. Afin d'augmenter le nombre de tests possibles, il a donc été choisi de limiter le temps de calcul.
- un fichier exécutable est ensuite lancé : il permet de compiler tous les fichiers .log dans un seul et même document. Les résultats sont classés par ordre croissant. Le fichier de résultats obtenu est ensuite importé dans le tableur Excel™.
- nous traitons les scores obtenus pour le score de similarité 2 car ce sont les bielles en base que nous cherchons à reconnaître dans la bielle scannée. Ce score est alors coloré en rouge et mis en gras ainsi que la signature du deuxième jeu correspondante. Pour rappel, pour chaque comparaison entre deux signatures, nous obtenons trois scores de similarité (cf. section 3.4).

Le tableau 4.14 illustre les résultats obtenus pour la bielle scannée avec 6 clusters ; les autres tableaux étant en annexe B.1. La donnée en base de connaissances ayant obtenu le score le plus élevé pour la similarité 2 est la bielle de 10 clusters avec un score de 80%.

2. C'est une machine virtuelle, configurée avec un processeur Intel Xeon Core 2 / 2.40 GHz avec 8.00 Go de RAM.

6 clusters				
Signature1 (1er jeu de données)	**Signature2 (2eme jeu de données)**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_100clusters.xml;	100	6	11
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_090clusters.xml;	100	6	12
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_080clusters.xml;	100	7	13
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_070clusters.xml;	100	8	15
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_060clusters.xml;	100	10	18
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_050clusters.xml;	100	12	21
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_040clusters.xml;	100	15	26
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_035clusters.xml;	100	17	29
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_030clusters.xml;	100	20	33
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_025clusters.xml;	100	24	38
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_020clusters.xml;	100	30	46
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_015clusters.xml;	100	40	57
signature_niv1_scan_bielle_ifpen_006clusters.xml;	**signature_niv1_bielle_ass_005clusters.xml;**	66	**80**	72
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_010clusters.xml;	100	60	75

Tableau 4.14 – Résultats de la comparaison pour une bielle scannée et segmentée avec 6 clusters.

Le tableau ci-dessous résume l'ensemble des résultats obtenus pour la bielle scannée de 6 à 10 clusters (voir détails des résultats en annexe B.1).

Nombre de clusters pour la donnée scannée (1er jeu)	6	7	8	9	10
Meilleur score de similarité 2 obtenu	80%	100%	100%	100%	100%
Nb de clusters de la bielle en base avec le meilleur score (2e jeu)	5	5	5	5	5

Nous pouvons remarquer que seules les signatures avec un nombre de clusters faible (5 exactement) obtiennent les meilleurs scores de similarité. Si nous regardons maintenant le meilleur score de similarité 3 (similarité globale) dans le tableau de B.1 ce qui correspond à chaque dernière ligne des tableaux, les données correspondantes n'ont guère un nombre de clusters plus important (10 ou 15 clusters au maximum).

Jeu de données pour le vilebrequin :

— 1er jeu de données : pour le vilebrequin scanné, nous avons 4 données avec 5, 10, 15 et 20 clusters.
— 2ème jeu de données : nous avons un ensemble de vilebrequins issus de la CAO de 5 à 100 clusters.

Le principe de test est similaire à celui de la bielle : nous avons comparé chaque donnée du premier jeu avec l'ensemble du deuxième jeu. Pour le vilebrequin avec 5 clusters, nous retrouvons les résultats dans le tableau 4.15 ; le reste des résultats étant en annexe dans le tableau B.2.

5 clusters				
Signature1 (1er jeu de données)	**Signature2 (2eme jeu de données)**	**Similarité1**	**Similarité2**	**Similarité3**
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_100clusters.xml	100	5	9
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_090clusters.xml	100	5	10
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_095clusters.xml	100	5	10
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_080clusters.xml	100	6	11
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_085clusters.xml	100	5	11
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_075clusters.xml	100	6	12
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_070clusters.xml	100	7	13
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_065clusters.xml	100	7	14
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_060clusters.xml	100	8	15
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_055clusters.xml	100	9	16
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_050clusters.xml	100	10	18
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_045clusters.xml	100	11	20
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_040clusters.xml	100	12	22
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_035clusters.xml	100	14	25
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_030clusters.xml	100	16	28
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_025clusters.xml	100	20	33
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_020clusters.xml	100	25	40
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_015clusters.xml	100	33	50
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_010clusters.xml	100	50	66
signature_scan_vilebrequin_005clusters.xml	**signature_vilebrequin_bdd_005clusters.xml**	80	**80**	80

Tableau 4.15 – Résultats de la comparaison pour vilebrequin scanné et segmenté avec 5 clusters.

En observant les scores obtenus pour la similarité 2, nous retenons un score de 80% pour le vilebrequin avec 5 clusters. En prenant également les résultats disponibles en annexe, nous avons le tableau de synthèse suivant :

Nombre de clusters pour la donnée scannée (1er jeu)	5	10	15	20
Meilleur score de similarité 2 obtenu	80%	100%	100%	100%
Nb de clusters du vilebrequin en base avec le meilleur score (2e jeu)	5	5	5	5

Les résultats obtenus ne sont guère plus concluants que les précédents avec la bielle. En effet, seule la donnée avec 5 clusters “matche” avec un score de 100% (ou proche). Dans notre idéal, une donnée scannée avec un certain nombre de clusters aurait dû “matcher” (avec un score maximal) avec une donnée du deuxième jeu ayant à peu près le même nombre de clusters. De plus, pour ce deuxième test, le deuxième jeu de données (issu de la CAO) a dû être réduit : les calculs de comparaison de certaines signatures ont été interrompus grâce à la minuterie.

Lorsque nous observons le graphe du vilebrequin avec 5 clusters (en base) sur la figure 4.16, nous remarquons sa simplicité (seulement 5 arêtes pour 5 nœuds), d’où sa facilité à “matcher” avec de nombreux graphes. En inspectant un des fichiers de résultat de la comparaison (avec le vilebrequin scanné), nous pouvons constater un grand nombre de correspondances possibles (supérieur à 20) entre les deux graphes. Nous pouvons ainsi imaginer qu’un grand nombre d’autres graphes pourrait “matcher” avec celui-ci.

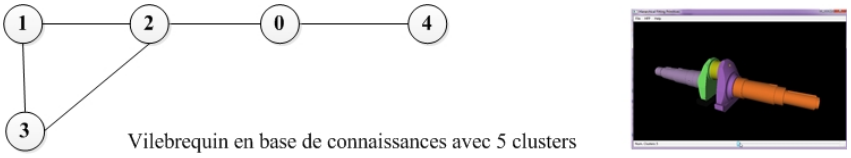


FIGURE 4.16 – Signature de niveau 1 d’un vilebrequin en base de connaissances (avec 5 clusters).

Maintenant si nous examinons les résultats obtenus pour la similarité 3 (3ème colonne) de l'ensemble des tableaux (voir annexe B.2), nous remarquons que les résultats obtenus sont assez différents : le vilebrequin 20 clusters scanné "matche" avec un vilebrequin 25 clusters de la base de connaissances et ceci pour un score 55%. En effet, sur la figure 4.17, nous pouvons remarquer que les deux segmentations sont quasiment identiques (même délimitation entre les surfaces), ce qui pourrait impliquer des graphes similaires, or le score n'est seulement que de 55%.

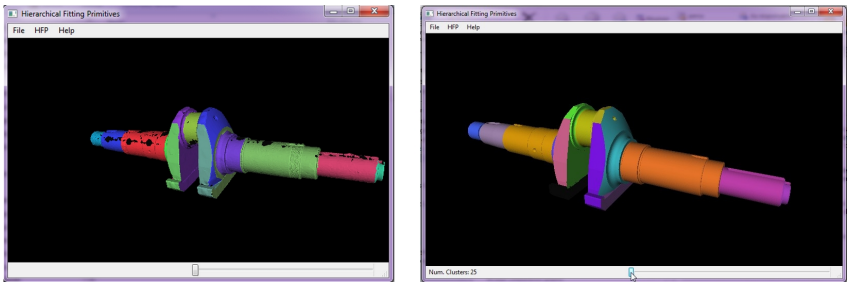


FIGURE 4.17 – Comparaison de deux segmentations entre un vilebrequin scanné (20 clusters) à gauche et un vilebrequin issu d'une donnée CAO (25 clusters) à droite.

Cependant ces remarques ne s'appliquent pas au cas de la bielle. En effet, pour les bielles scannées avec 7, 8 et 9 clusters, le score de similarité 2 le plus élevé correspond aussi au score de similarité 3 le plus haut (la bielle avec 5 clusters restant identifiée comme composant le plus similaire à la donnée scannée).

Grâce à ces premiers tests, nous pouvons faire les conclusions suivantes pour le niveau 1 :
- la comparaison des données avec un nombre de clusters supérieur à 20 pour les données scannées est impossible ;
- le faible niveau d'informations apporté par le graphe de niveau 1 pourrait laisser la possibilité d'obtenir des composants identifiés comme similaires alors qu'ils ne le sont pas.

Pour cela, d'autres tests sur un échantillon plus large de composants seront réalisés et présentés dans la section 4.4.

4.3.2 Comparaison de niveau 2 pour la bielle et le vilebrequin

Nous avons commencé avec les signatures de niveau 2 pour la bielle.

Jeux de données :
- 1er jeu de données : pour la bielle scannée le nombre de clusters a pu être augmenté. Le jeu comporte désormais les signatures de niveau 2 avec les valeurs suivantes : 5, 10, 15 et 20 clusters.
- 2ème jeu de données : pour la bielle issue de la CAO, nous avons utilisé 14 signatures de niveau 2 comprises entre 5 et 100 clusters.

Le principe des tests reste similaire : seule, la comparaison évolue puisque les attributs (type de cluster) des nœuds sont pris en compte.

L'intégralité des résultats est en annexe dans les tableaux B.3. Le tableau ci-dessous reprend les résultats obtenus pour ce deuxième niveau de signature.

Nb de clusters pour la donnée scannée (1er jeu)	5	10	15	20
Meilleur score de similarité 2 obtenu	80%	80%	100%	100%
Nb de clusters de la bielle en base avec le meilleur score (2e jeu)	5	5	5	5

Nous observons que c'est la bielle en base avec 5 clusters qui obtient systématiquement le score de similarité 2 le plus élevé. Nous allons regarder de plus près la comparaison de ces deux signatures et analyser leur *mapping*. La figure 4.18 illustre les résultats de la comparaison de ces deux signatures (avec le *mapping* entre les deux graphes en gras).

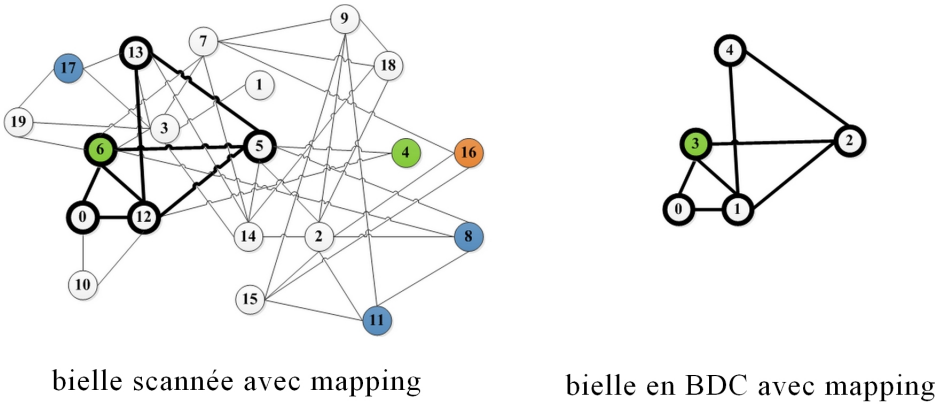


Figure 4.18 – Résultats de la comparaison de niveau 2 de la bielle scannée (20 clusters) avec une bielle (5 clusters) de la base de connaissances.

Nous pouvons observer qu'un seul nœud de type "Sphère" fait partie du *mapping* et que les quatre autres nœuds sont de type "Autre". Si nous regardons à nouveau le tableau de résultats B.3. Le deuxième meilleur score de similarité 2 obtenu est de 70% pour une bielle (CAO) avec 10 clusters, puis en troisième, nous trouvons un score de 60% pour une bielle avec 15 clusters. La figure 4.19 illustre les deux graphes relatifs à ce troisième score, avec en gras, un des *mapping* obtenus. Nous souhaitons nous intéresser à cette donnée en base car c'est celle qui, dans notre idéal, aurait dû "matcher" avec un des scores les plus élevés.

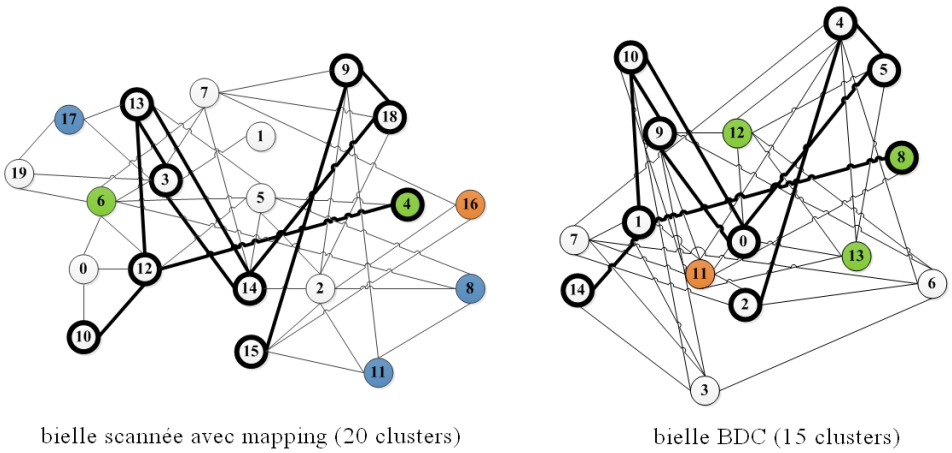


Figure 4.19 – Comparaison entre deux signatures de niveau 2 : une bielle scannée avec 20 clusters comparée à une bielle en base avec 15 clusters. Un des *mapping* est représenté en gras.

Grâce à l'affichage du graphe, nous observons à nouveau qu'un seul nœud est typé sur les 9 identifiés dans le *mapping*. Pour finir, nous allons observer le *mapping* entre notre bielle scannée (20 clusters) et la bielle en base ayant 20 clusters pour lequel le score de similarité obtenu est de 55%. Cette donnée correspond également au meilleur score obtenu pour la similarité globale (dernière ligne et dernière colonne du tableau). La figure 4.20 illustre cela.

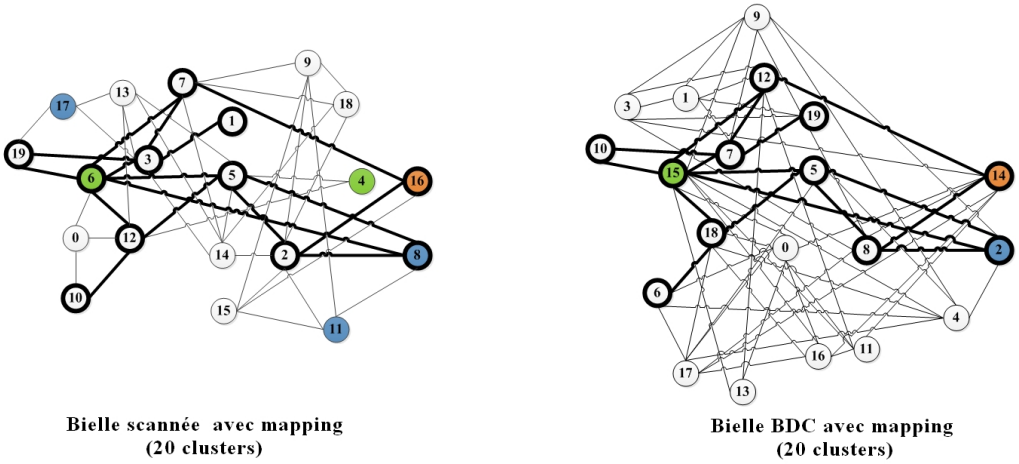


FIGURE 4.20 – Comparaison entre deux signatures de niveau 2 : une bielle scannée avec 20 clusters et une bielle en base de connaissances avec 20 clusters. Un des *mapping* entre les deux graphes est représenté en gras.

En étudiant le *mapping* obtenu lors de cette dernière comparaison (figure 4.20), nous observons que 3 clusters sur 11 sont typés dont une sphère, un plan et un cylindre ; tous les autres étant de type "Autre".

De ces comparaisons de signatures de la bielle scannée avec 20 clusters, nous pouvons déduire que lorsque la similitude entre deux signatures diminue, le niveau d'informations topologiques croît. Le tableau ci-dessous résume les valeurs permettant d'arriver aux déductions sus-citées.

Score S2	Nombre de clusters de la donnée en BDC	Nombre de surfaces typées du *mapping*
100%	5	1
60%	15	1
55%	20	3

Pour la troisième ligne du tableau ci-dessus, nous avons ainsi un score de similarité de 55% entre une bielle scannée avec 20 clusters et une bielle en base (20 clusters également). Il en résulte un *mapping* comportant 11 nœuds dont 3 qui sont typés.

Nous pouvons donc préconiser des améliorations possibles à l'algorithme de comparaison en pondérant le score de similarité par un ratio (par exemple : le nombre de surfaces typées dans le *mapping* par rapport au nombre total de nœuds du *mapping*). L'idée serait que le nombre de surfaces typées présentes dans le mapping puissent influer sur le score de similarité.

Maintenant, nous allons regarder les résultats obtenus en comparant le vilebrequin scanné à l'ensemble des vilebrequins de la base de connaissances afin d'infirmer ou confirmer ces premières conclusions.

Jeux de données pour le vilebrequin :

- 1er jeu de données : pour le vilebrequin scanné le nombre de clusters a pu également être augmenté. Le jeu comporte désormais les signatures de niveau 2 avec les valeurs suivantes : 5, 10, 15, 20 etc. jusqu'à 65 clusters.
- 2ème jeu de données : pour la bielle issue de la CAO, nous avons utilisé 20 signatures de niveau 2 comprises entre 5 et 100 clusters.

L'ensemble des résultats est présenté dans les tableaux B.4, B.5, B.6 et B.7 en annexe.

De la même manière, nous proposons les tableaux de synthèse ci-dessous :

Nb de clusters pour la donnée scannée	5	10	15	20	25	30
Meilleur score de similarité 2 obtenu	80%	100%	100%	100%	100%	100%
Nb de clusters du vilebrequin en base avec le meilleur score	5	5	5	5	5	5

Nb de clusters pour la donnée scannée	35	40	45	50	55	60	65
Meilleur score de similarité 2	100%	80%	100%	80%	80%	80%	50%
Nb de clusters du vilebrequin en base	5	5	5	5	5	5	20

Nous remarquons que pour les données scannées entre 5 et 35 clusters, les résultats obtenus sont similaires à ceux obtenus pour les tests avec les bielles : c'est à nouveau la donnée en base avec 5 clusters qui obtient un score de 100% pour la similarité 2. A partir de 40 clusters, le score de similarité diminue. Nous observons également que pour le dernier test avec 65 clusters, le meilleur score retenu correspond à un vilebrequin avec 20 clusters.

De la même manière, nous pouvons également mettre en corrélation ces valeurs avec le nombre de clusters typés dans les fichiers de *mapping*. Pour cela, nous avons choisi de mettre en exergue les valeurs obtenues pour les données scannées avec 20, 40 et 65 clusters dans le tableau ci-après. Les valeurs obtenues pour la troisième colonne ont été obtenues en étudiant les fichiers de résultats (.log) que nous avons ensuite mis en relation avec les fichiers de signatures pour retrouver le type de chaque nœud du *mapping*.

Score S2	Nombre de clusters de la donnée en BDC	Nombre de surfaces typées du *mapping*
100%	20	0
80%	40	0
50%	65	4

Pour la première ligne du tableau ci-dessus, nous obtenons une similarité de 100% entre un vilebrequin scanné avec 20 clusters et un vilebrequin en base avec 5 clusters. De cette comparaison, il en résulte un *mapping* constitué de 5 nœuds de type "Autre". Nous pouvons alors considérer que la comparaison de niveau 2 n'est pas satisfaisante car son *mapping* ne peut fournir aucune information topologique. En effet, les nœuds identifiés comme similaires (car de type "Autre") ne sont peut être pas similaires dans la réalité. Par exemple, une surface conique et une surface de forme libre pourraient être "similaires" car toutes deux seraient non-typées.

A l'étude du fichier .log issu de la comparaison, nous remarquons également que le nombre de correspondances (nombre de *mapping*) a été réduit considérablement par rapport au nombre de correspondances obtenues lors des comparaisons de niveau 1 avec la donnée en base de 5 clusters.

Pour la deuxième ligne du tableau, nous avons une similarité de 80% entre une donnée scannée de 40 clusters et une donnée en base avec 5 clusters. Le mapping qui en découle contient que 4 nœuds de type "Autre". Ce résultat n'est pas satisfaisant également.

Enfin **pour la dernière ligne**, nous obtenons une similarité de 50% entre le vilebrequin scanné avec 65 clusters et le vilebrequin en base avec 20 clusters. Le mapping entre ces deux graphes comporte 9 nœuds dont 3 qui sont de type "Plan" et un de type "Sphère".

Les premières déductions faites avec la comparaison de la bielle se confirment avec le vilebrequin. En effet, nous observons que lorsque la similarité entre les deux signatures diminue, le niveau d'information fourni par cette comparaison croît. Ces derniers tests le montrent de manière significative car seuls les meilleurs scores de similarité 2 ont été utilisés.

La préconisation faite précédemment de pondérer le score de similarité en fonction du nombre de nœuds typés dans le mapping est ainsi confirmée.

4.3.3 Étude des coûts de calcul pour les comparaisons de niveaux 1 et 2

Dans cette section, nous allons nous intéresser au temps de calcul de comparaison pour le niveau 1 puis pour le niveau 2.

Jeux de données utilisés :
- 1er jeu de données : il contient un assemblage scanné d'un piston et d'une bielle ;
- 2ème jeu de données : il contient 5 maillages dont 3 assemblages.

La figure 4.21 représente ces deux jeux.

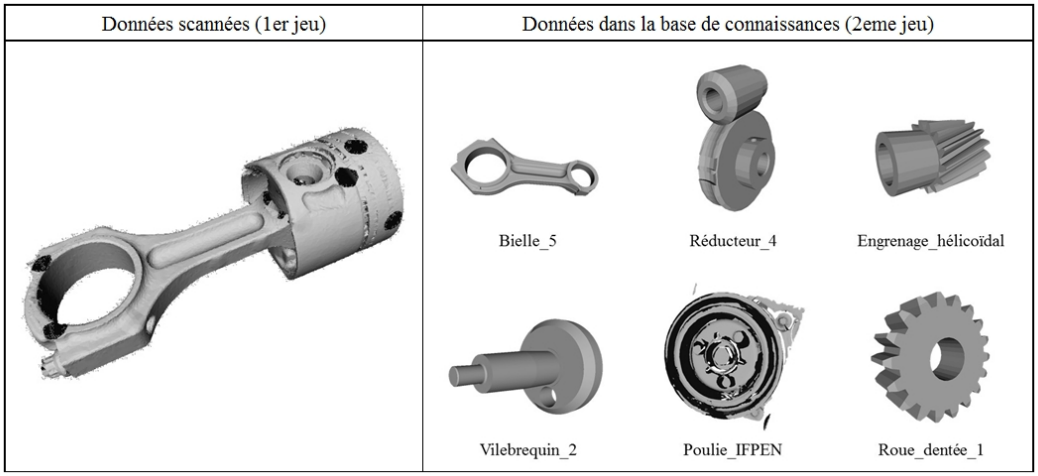


FIGURE 4.21 – Ensemble de données testées pour l'évaluation du coût de comparaison (temps de calcul).

Variables :
- Variable 1 : le nombre de clusters des signatures du premier jeu varie de 15 à 50 clusters (15, 20, 25 etc. jusqu'à 50 clusters) et ceci pour les deux niveaux de signature.
- Variable 2 : le nombre de clusters du deuxième jeu évolue également. Les signatures de niveau 1 et 2 ont été générées pour 15, 20 et 25 clusters.

Principe du test : Les opérations suivantes ont été réalisées :
- chaque donnée de X clusters du premier jeu est comparée à l'ensemble des données du deuxième jeu dont le nombre de clusters a été fixé (15 clusters pour commencer) ;
- une minuterie est définie et permet de stopper le calcul de comparaison au bout de 5h ;
- les scores de similarité obtenus pour chaque donnée dont le calcul n'a pas été stoppé, sont inscrits dans un fichier de synthèse .log ;
- ce document de synthèse est importé dans un tableur Excel™ ;
- l'heure de création/modification de chaque fichier .log de comparaison est utilisée pour définir approximativement le temps de calcul. L'heure et la date annotées au fichier correspondent au dernier enregistrement. Nous faisons ensuite la soustraction entre les heures d'enregistrement des différents fichiers afin de définir le temps de calcul pour chaque comparaison ;
- en fonction du temps calculé, une couleur est attribuée selon l'échelle définie dans la figure 4.22.

Figure 4.22 – Code couleur utilisé pour évaluer le coût de comparaison entre deux signatures

— la variable 1 est augmentée de 5 clusters et les calculs sont relancés avec le même deuxième jeu ;
— les tests sont stoppés dès lors qu'il est impossible de continuer à augmenter la variable 1. Lorsque plusieurs tests ont obtenu un temps de calcul supérieur à 5h, nous stoppons la série de tests ;
— la série suivante démarre avec les données suivantes : variable 1 = 15 clusters (donnée scannée) et variable 2 = 20 clusters (pour les données du deuxième jeu).

Matériel utilisé : l'intégralité des tests de comparaison ont été lancés sur une machine virtuelle dédiée et configurée avec un processeur Intel Xeon Core 2 / 2.40 GHz avec 8.00 Go de RAM.

4.3.3.1 Étude des coûts de calcul pour le niveau 1

Pour variable 2 = 15 clusters : Nous obtenons le tableau de résultats 4.16, le reste des résultats étant dans les tableaux en annexes B.8.

15 clusters					
Signature1 (1er jeu de données)	**Signature2 (2eme jeu de données)**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_scan_bielle_piston_15clusters.xml	signature_niv1_bielle_5_15clusters.xml	53	53	53	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_engrenage_helicoidal_15clusters.xml	60	60	60	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_roue_dentee_1_15clusters.xml	60	60	60	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_poulie_ifpen_15clusters.xml	66	66	66	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_vilebrequin_2_15clusters.xml	66	66	66	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_reducteur_4_15clusters.xml	73	73	73	< 1 min

Tableau 4.16 – Estimation du coût de comparaison d'un ensemble de données (avec 15 clusters) pour le niveau 1.

Nous observons que pour ces premiers tests le temps de calcul pour effectuer chaque comparaison de graphes est inférieur à une minute. La case est alors coloriée en vert et le test suivant peut être lancé. Cela consiste à augmenter le nombre de clusters de la signature 1 (assemblage "scan bielle piston" avec un nombre de clusters de 20). La signature de niveau 1 correspondante est alors comparée avec toutes les signatures 2, c'est-à-dire du deuxième jeu (bielle_5, engrenage_hélicoïdal, roue_dentée etc.). Nous retrouvons ces résultats dans le tableau B.8 cité précédemment.

Pour variable 2 = 20 clusters : Les premiers résultats sont dans le tableau 4.17 et le reste dans le tableau B.9 en annexe.

15 clusters					
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_scan_bielle_piston_15clusters.xml	signature_engrenage_helicoidal_20clusters.xml	60	45	51	< 30 min
signature_scan_bielle_piston_15clusters.xml	signature_roue_dentee_1_20clusters.xml	60	45	51	< 2 min
signature_scan_bielle_piston_15clusters.xml	signature_bielle_5_light_20clusters.xml	73	55	62	< 2 min
signature_scan_bielle_piston_15clusters.xml	signature_poulie_ifpen_20clusters.xml	73	55	62	< 15 min
signature_scan_bielle_piston_15clusters.xml	signature_reducteur_4_20clusters.xml	73	55	62	< 2 min
signature_scan_bielle_piston_15clusters.xml	signature_vilebrequin_2_20clusters.xml	80	60	68	< 20 min

Tableau 4.17 – Estimation du coût de comparaison d'un ensemble de données (avec 20 clusters) pour le niveau 1.

Nous remarquons que pour cette deuxième série de tests (variable 2 = 20 clusters), le temps de calcul est supérieur par rapport aux premiers tests. Par exemple pour l'engrenage hélicoïdal, le temps de comparaison des deux graphes est d'environ 30 minutes (soit 30 fois plus que précédemment).

Pour variable 2 = 25 clusters : Les premiers résultats sont dans le tableau 4.18 et le reste dans le tableau B.10 en annexes.

15 clusters					
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_engrenage_helicoidal_25clusters.xml	60	36	45	< 6 min
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_roue_dentee_1_25clusters.xml	60	36	45	< 4 min
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_poulie_ifpen_25clusters.xml	73	44	55	< 5 min
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_reducteur_4_25clusters.xml	80	48	60	< 1h22
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_bielle_5_light_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_vilebrequin_2_25clusters.xml				> 5h

Tableau 4.18 – Estimation du coût de comparaison d'un ensemble de données (avec 25 clusters) pour le niveau 1.

Nous remarquons qu'à partir de cette troisième série de tests (variable 2 = 25 clusters), certaines comparaisons deviennent impossible à réaliser notamment en ce qui concerne la bielle et le vilebrequin (signature 2). Pour ces deux données, les scores n'ont pas été affichés car le calcul du meilleur *mapping* n'a pas abouti. Les tests ont d'ailleurs été stoppés (test avec variable 2 = 30 clusters non effectué).

Nous observons que le nombre maximal de clusters pour la donnée scannée (variable 1) ne dépasse pas 25 clusters tous tests confondus pour le premier niveau de comparaison de graphes.

Nous allons maintenant reproduire ces tests sur les signatures de niveau 2.

4.3.3.2 Étude des coûts de calcul pour le niveau 2

Pour ce test, les jeux de données utilisés, les variables, le principe du test et le matériel utilisé restent identiques aux tests pour le premier niveau. Seuls les fichiers de signature de niveau 1 sont remplacés par ceux de signature de niveau 2 et ceci pour les deux jeux de données.

L'ensemble de ces tests étant assez répétitif, nous allons directement présenter les premiers tableaux obtenus (variable 1 = 15 clusters) pour chaque série de tests (variable 2 = { 15, 20, 25 }) dans le tableau 4.19.

1ère série

15 clusters					
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	33	33	33	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_reducteur_4_15clusters.xml	46	46	46	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_bielle_5_15clusters.xml	53	53	53	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	60	60	60	< 10s

2eme série

15 clusters					
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_reducteur_4_20clusters.xml	26	20	22	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	33	25	28	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	53	40	45	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	73	55	62	< 10 s

3eme série

15 clusters					
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	26	16	20	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_reducteur_4_25clusters.xml	40	24	30	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	60	36	45	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_bielle_5_light_25clusters.xml	80	48	60	< 10 s

Tableau 4.19 – Estimation du coût de comparaison d'un ensemble de données pour le niveau 2.

En observant ces résultats, nous constatons une baisse singulière du coût de comparaison qui ne dépasse pas les 10 secondes pour chaque série de tests.

Le reste des résultats est présenté en annexes dans les tableaux suivants :
— pour variable 2 = 15 clusters : Tableau B.11 et Tableau B.12
— pour variable 2 = 20 clusters : Tableau B.13 et Tableau B.14
— pour variable 2 = 25 clusters : Tableau B.15 et Tableau B.16

Nous notons également que le nombre maximal pour la variable 1 atteint les 50 clusters, ce qui correspond quasiment au double des capacités comparé aux tests pour le niveau 1.

En calculant la moyenne des temps obtenus pour l'ensemble de données suivant : variable 1 = 15,20,25,30 et variable 2 = 15 clusters (soit les valeurs des tableaux B.8 et B.11), nous obtenons M1 = 7378 secondes et M2 = 18 secondes avec M1 la moyenne calculée pour le niveau 1 et M2 pour le niveau 2. Nous pouvons ainsi déduire pour cette série de tests, que le temps moyen de comparaison de niveau 1 est environ 410 fois supérieur à celui pour le niveau 2.

De la même manière, nous avons calculé ces moyennes pour variable 2 = 20 clusters (valeurs des tableaux B.9 et B.13). Nous obtenons M1 = 11 153 secondes et M2 = 1252 secondes, soit un coefficient d'environ 9 entre les deux moyennes.

Enfin pour variable 2 = 25 clusters (valeurs des tableaux B.10 et B.15), nous avons M1 = 9868 secondes M2 = 2373 secondes soit un coefficient d'environ 4 entre les deux moyennes.

Nous constatons alors que plus le nombre de clusters augmente pour la variable 2, plus la différence entre le temps de comparaison entre niveau 1 et niveau 2 diminue (diminution de la valeur des coefficients calculés).

Au vu de ces résultats, nous pouvons clairement mettre en évidence que la comparaison de graphe pour le niveau 1 requiert un temps de calcul bien supérieur (jusqu'à 410 fois) par rapport au niveau 2.

Conclusion sur les expérimentations réalisées

Grâce aux expérimentations, nous pouvons mettre en exergue les points suivants :

— la valeur du seuil influe sur le nombre de clusters nécessaires pour obtenir une signature de niveau 2 qui soit exploitable. Ce sont les types de clusters qui la caractérisent, d'où l'importance d'ajuster le maximum de clusters dans la donnée signée. Plus la valeur du seuil de détection (pour la signature de niveau 2) est faible, plus il faut un nombre important de clusters afin d'avoir plusieurs clusters typés.

— après expérimentation de la comparaison par graphe et en particulier des capacités de calcul, il est quasiment impossible de comparer des graphes de grande taille (de plus de 50 nœuds) avec le matériel actuel (tests sur d'autres machines). Par expérience, nous préconiserons alors l'utilisation d'un seuil entre 1,5 et 2 mm afin d'obtenir une signature de niveau 2 avec un niveau de détails convenable (pourcentage de surfaces ajustées supérieur à 50% pour une segmentation d'environ 40 clusters pour les trois données scannées testées).

— dans un contexte d'utilisation de notre méthodologie, il sera préférable de limiter le temps de comparaison de graphes à 5 minutes afin de pouvoir comparer autant de signatures que possible.

— la majorité des tests ont été réalisés en comparant des signatures de données scannées avec des signatures de données non-scannées (CAO). Ceci s'explique par le fait que nous n'avions pas suffisamment de données scannées à notre disposition. L'utilisation routinière de la méthodologie HDI-RE permettrait d'augmenter le nombre de signatures issues de scan dans la base de connaissances. Nous pouvons ainsi imaginer que cette étape de capitalisation pourrait améliorer les résultats obtenus notamment pour la signature par critère iso-périmétrique ou encore les signatures par graphe.

— des améliorations sont encore nécessaires notamment pour le calcul de similarité (pondération du score en fonction du nombre de surfaces typées dans le *mapping*), ou encore dans l'algorithme de détection des cylindres pour l'étape de signature.

4.4 Application à un cas industriel

Dans cette section, nous présenterons un cas d'étude auquel nous allons appliquer notre méthodologie HDI-RE. L'ensemble de ce test a été réalisé en utilisant étape par étape, les algorithmes et solutions logicielles présentés dans notre chapitre proposition.

Hypothèses :

1. L'algorithme de signature par graphe de niveau 3 n'ayant pas été développé, cette étape sera réalisée manuellement.
2. Seules les données de type "nuage de points / maillage" seront utilisées pour l'application de notre méthodologie.
3. Aucune valeur minimale (en %) n'a été choisie pour retenir un résultat par rapport à son score de similarité. Ainsi pour l'ensemble des résultats obtenus, plusieurs possibilités s'offriront à l'utilisateur : choix du ou des meilleurs scores (proches de 100%) pour considérer qu'une signature est similaire à la donnée d'entrée (c'est-à-dire que certains composants ont été identifiés à X% dans la donnée d'entrée). Autrement, l'utilisateur pourra choisir de retenir une autre signature avec un score de similarité plus faible s'il juge plus pertinente la signature considérée par rapport à celles qui auraient obtenues un score de similarité maximal. Ce sera notamment le cas au niveau de la signature de niveau 2, si l'utilisateur souhaite continuer avec le niveau 3. En effet, il aura tout intérêt à choisir une signature de niveau 2 avec un nombre de clusters élevé pour maximiser le nombre de clusters typés présents dans les *mapping*.
4. Nous ferons l'hypothèse que chaque signature de niveau 2 de la base de connaissances est reliée à une signature de niveau 3. Cette dernière étant générée en dehors du notre processus de RE, nous n'étudierons pas sa pertinence. En d'autres termes, pour des signatures de niveau 2 avec un nombre de clusters faible (5 clusters par exemple), leur signature de niveau 3 associée risque d'être pauvre en "information fonctionnelle". En effet si le nombre de clusters est faible, alors plusieurs "surfaces" seront fusionnées et elles ne pourront être associées à des surfaces fonctionnelles. Ainsi certains nœuds du graphe de précédence ne pourront être mis en relation avec la signature de niveau 2 (construction du graphe GP_{fus}).

Contenu de la base de connaissances : Nous considérons que notre base de connaissances contient toutes les signatures possibles (globale et par graphe) de tous les composants testés précédemment hormis l'assemblage testé dans cette section (données d'entrée). La figure 4.23 présente toutes les pièces qui correspondent aux signatures présentes en base.

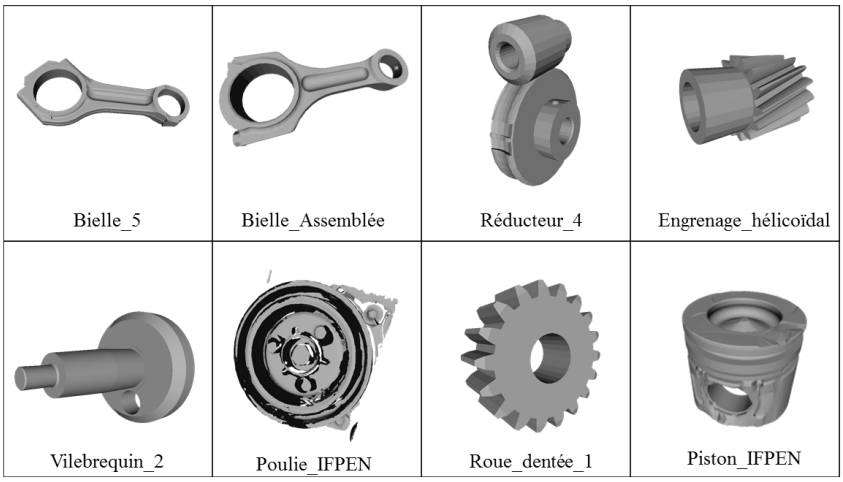


Figure 4.23 – Nuages de points des données correspondants aux signatures présentes dans la base de connaissances dans le cadre de ce test.

Objectifs du test : Le premier objectif est d'identifier les composants présents dans les données d'entrée et en particulier de reconnaître le "Piston_IFPEN" ainsi que la "Bielle_Assemblée" (voir figure 4.23). L'objectif final (en sortie de notre processus de RE) est de reconstruire une maquette numérique qui soit paramétrable. De plus, nous sommes dans le cas d'étude "from scratch", où aucune maquette initiale de notre assemblage n'est fournie comme donnée d'entrée.

Ainsi afin de parvenir en sortie à une maquette numérique avec des modèles CAO paramétrables, il nous faut extraire les informations géométriques et topologiques des composants identifiés dans les données d'entrée. Cela passera également par l'identification de surfaces fonctionnelles (signature de niveau 3).

Jeu de données testé : Les données d'entrée ont été fournies par l'Institut Français du Pétrole et des Énergie Nouvelles (IFPEN), partenaire du projet METIS sur lequel reposent ces travaux. Il s'agit d'un assemblage scanné avec une bielle (composée de deux parties) et d'un piston ainsi que tous les éléments de fixation (visserie). Nous disposons également d'une photographie de l'assemblage. Cet ensemble de composants appartient à une automobile de marque Peugeot (moteur PSA 1.6 HDI 110 ch.). Les données 3D ont été acquises à l'aide d'un bras laser de marque HandyScan^{TM}. Le maillage est enregistré sous le format STL (ASCII).

La figure 4.24 représente les données en entrée de notre processus : il s'agit d'une photographie de l'assemblage ainsi que son maillage.

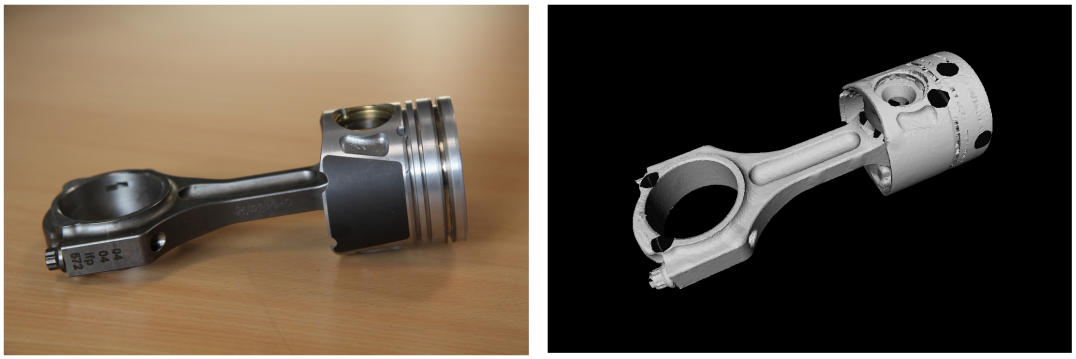


Figure 4.24 – Présentation des données d'entrée pour le cas d'étude : à gauche, une photographie de l'assemblage et à droite, le maillage de l'assemblage (les ronds noirs correspondent à des pastilles qui servent de repère lors de l'acquisition des données).

Principe du test : Nous appliquerons notre méthodologie HDI-RE à nos données d'entrée tel que décrit dans la figure 4.25. La segmentation des données sera présentée dans la sous-section 4.4.1. Puis au lieu de présenter séparément les étapes de signature et de comparaison comme effectué précédemment, nous allons les combiner pour suivre le cheminement suivant : signature niveau 1/comparaison niveau 1 dans la sous-section 4.4.2, signature niveau 2/comparaison niveau 2 dans la sous-section 4.4.3 et enfin signature niveau 3/comparaison niveau 3 dans la sous-section 4.4.4.

Remarque : Nous testerons la signature par critère iso-périmétrique à titre informatif (sous-section 4.4.5). En effet, cette signature ne fournit pas d'information géométrique, ce qui est nécessaire dans notre cas, afin de reconstruire une maquette numérique avec des modèles CAO paramétrés.

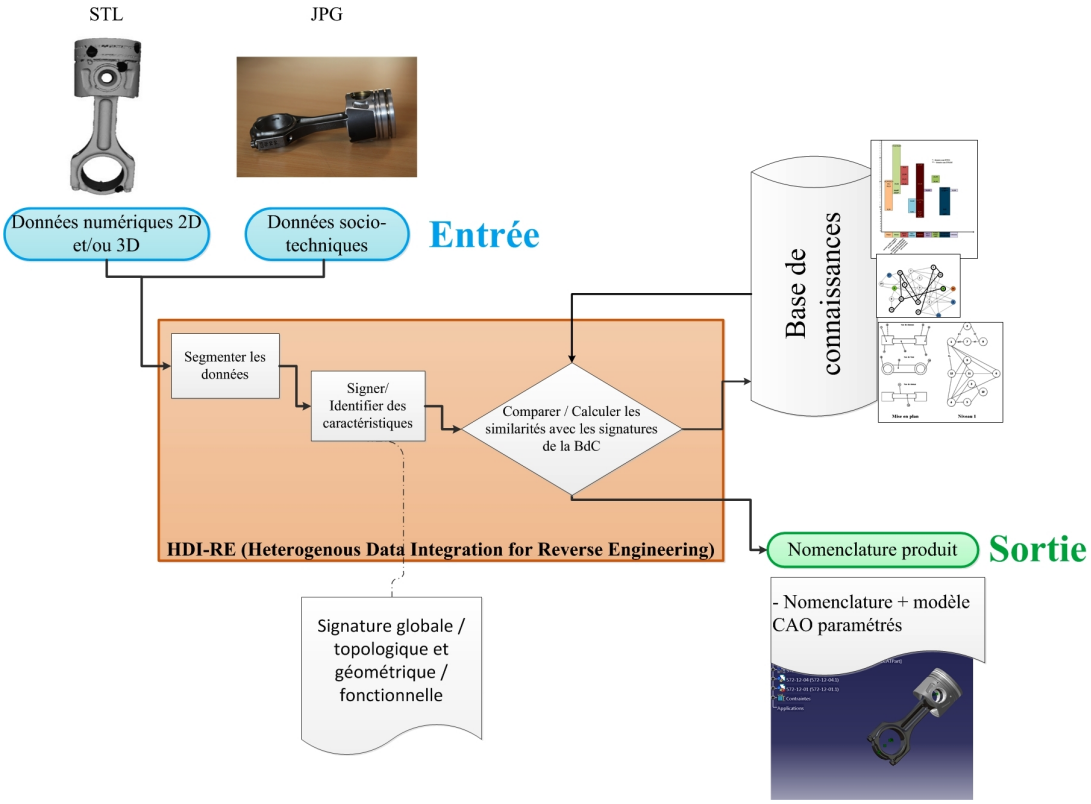


FIGURE 4.25 – Expérimentation de la méthodologie HDI-RE sur un ensemble de données.

4.4.1 La segmentation des données d'entrée

A l'aide du logiciel Efpisoft, nous avons segmenté le maillage issu du nuage de points. Pour cela, nous avons tout d'abord ouvert notre fichier STL et nous avons lancé la segmentation ("Run HPF"). Puis, nous avons déplacé la barre de défilement jusqu'à obtenir un nombre de clusters de 20. La figure 4.26 illustre ces opérations.

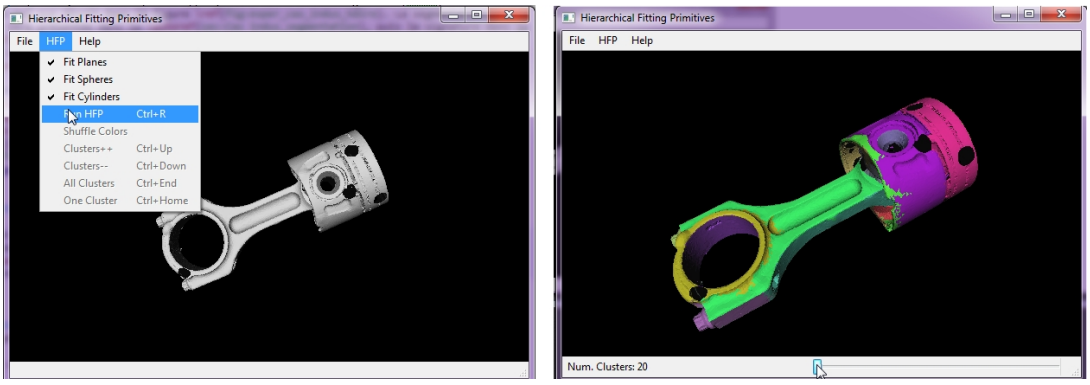


FIGURE 4.26 – Segmentation de l'assemblage avec le logiciel Efpisoft.

Une fois le nombre de clusters (segments) choisi, nous enregistrons notre "scène" sous le format .iv (*Open Inventor*). Ce fichier constitue la donnée en sortie de cette étape.

Le choix du nombre de clusters résulte des différents tests réalisés précédemment, il garantit la possibilité d'obtenir des résultats dans un laps de temps raisonnable (environ 2h pour l'étape de comparaison avec une base de données d'une vingtaine de signatures et tous niveaux de graphe confondus).

Pour cette expérimentation, nous avons choisi de suivre le scénario n°3 présenté dans la section 3.3.1. Cependant, nous présenterons également les résultats obtenus en suivant le scénario n°5 (afin de comparer les résultats de comparaison obtenus en sortie du niveau 2). La figure 4.27 rappelle les étapes qui seront présentées dans les prochaines sous-sections.

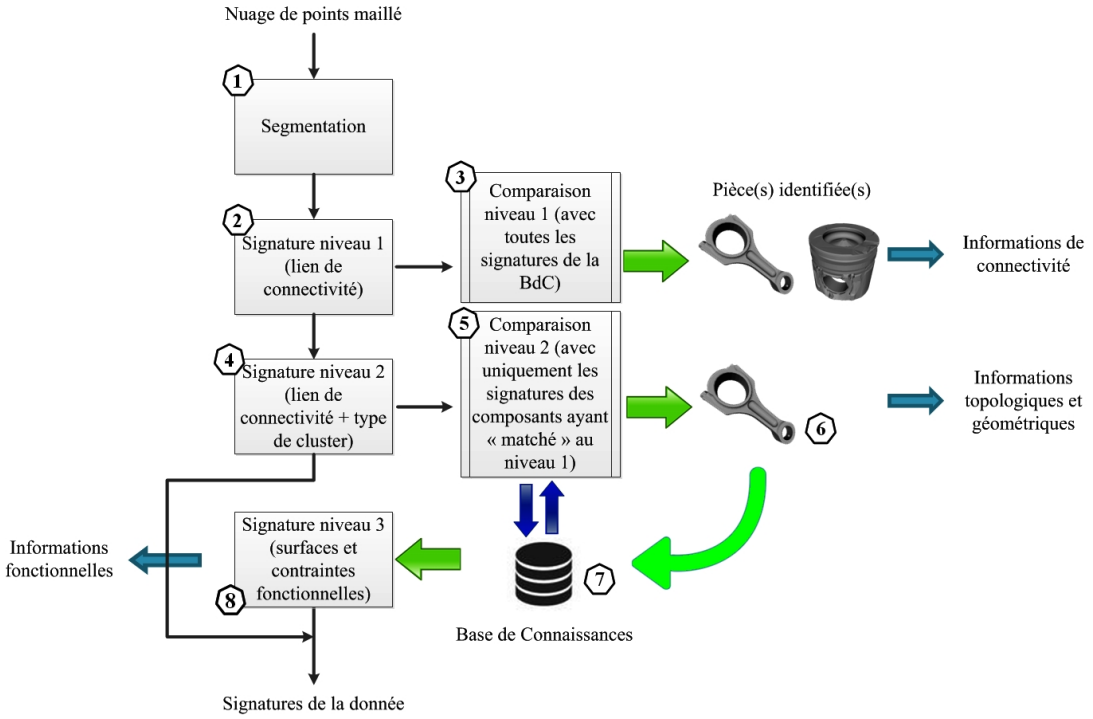


FIGURE 4.27 – Signature par graphe à trois niveaux (scénario n°3).

4.4.2 La signature de niveau 1 et sa comparaison

Après application de l'algorithme de signature de niveau 1 (étape n°2), nous obtenons le graphe illustré par la figure 4.28.

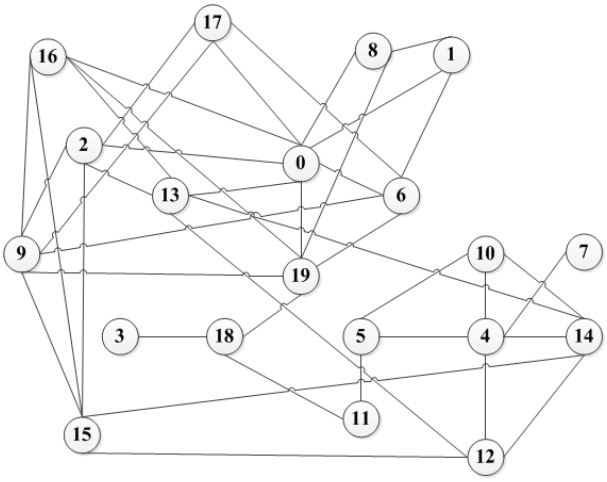


FIGURE 4.28 – Graphe de niveau 1 de l'assemblage scanné.

Pour l'étape de comparaison (étape n°3 de la figure 4.27), nous avons choisi de limiter le temps de calcul à 30 min par comparaison. En comparant le graphe de la figure 4.28 à l'ensemble des signatures de la base de connaissances et en gardant que les résultats dont le temps de comparaison est inférieur à 30 min, nous obtenons les résultats présentés dans le tableau 4.20.

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv1_data_entree_20clusters.xml	signature_niv1_bielle_5_15clusters.xml	45	60	51
signature_niv1_data_entree_20clusters.xml	signature_niv1_engrenage_helicoidal_15clusters.xml	55	73	62
signature_niv1_data_entree_20clusters.xml	signature_niv1_roue_dentee_1_15clusters.xml	55	73	62
signature_niv1_data_entree_20clusters.xml	signature_niv1_vilebrequin_2_15clusters.xml	55	73	62
signature_niv1_data_entree_20clusters.xml	signature_niv1_poulie_ifpen_15clusters.xml	60	80	68
signature_niv1_data_entree_20clusters.xml	signature_niv1_reducteur_4_15clusters.xml	60	80	68
signature_niv1_data_entree_20clusters.xml	signature_niv1_bielle_assemblee_10clusters.xml	40	80	53
signature_niv1_data_entree_20clusters.xml	signature_niv1_piston_ifpen_15clusters.xml	60	80	68
signature_niv1_data_entree_20clusters.xml	signature_niv1_piston_ifpen_10clusters.xml	45	90	60
signature_niv1_data_entree_20clusters.xml	**signature_niv1_bielle_assemblee_5clusters.xml**	25	**100**	40
signature_niv1_data_entree_20clusters.xml	**signature_niv1_piston_ifpen_5clusters.xml**	25	**100**	40

Tableau 4.20 – Résultats de la comparaison de niveau 1 avec la signature de l'assemblage scanné et l'ensemble de signatures en base.

En observant les résultats obtenus dans le tableau ci-dessus, nous constatons que les deux meilleurs scores obtenus (avec 100%) correspondent à une bielle et un piston avec 5 clusters chacun. Ces deux composants correspondent aux deux pièces à identifier (cf. objectifs du test). Nous remarquons également que le "piston_ifpen_10 clusters" obtient un score de 90%. Puis nous obtenons une similarité de 80% pour le "piston_ifpen_15 clusters", la "Bielle_Assemblée_10 clusters" mais aussi pour le "réducteur_4_15clusters" et la "poulie_IFPEN_15clusters". Ces deux derniers résultats sont étranges (score élevé alors que le composant est très différent). Nous allons alors étudier de plus près le *mapping* entre la signature de notre assemblage et la signature de ce réducteur. Sur la figure 4.29, nous pouvons observer en gras, un des *mapping* obtenus.

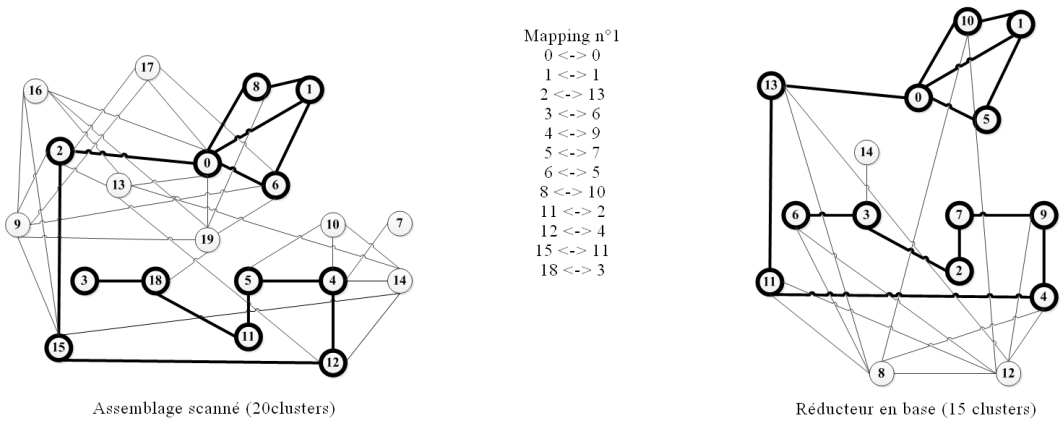


FIGURE 4.29 – Étude du mapping entre la signature de l'assemblage scanné et celle du réducteur (en gras, le plus grand sous-graphe commun).

Nous pouvons ainsi constater que la similarité entre ces deux graphes est importante, malgré leur différence géométrique.

L'étape suivante consiste à signer la donnée avec le deuxième niveau de graphe.

4.4.3 La signature de niveau 2 et sa comparaison

Pour la signature de niveau 2 (étape n°4), nous avons choisi une valeur de seuil de 1,5 mm identique à tous les types de clusters (plan, cylindre et sphère) afin de maximiser le nombre de clusters typés pour un nombre total de clusters minimal pour chaque signature. Comme nous avons pu le voir dans les conclusions de la section 4.2.1, pour un nombre de clusters donné (20 clusters par exemple), si nous réduisons les valeurs de seuil jusqu'à celles définies par l'étude expérimentale, alors le nombre de clusters typés diminue.

Avec les critères définis ci-dessus, nous obtenons le graphe pour la signature de niveau 2 dans la figure 4.30.

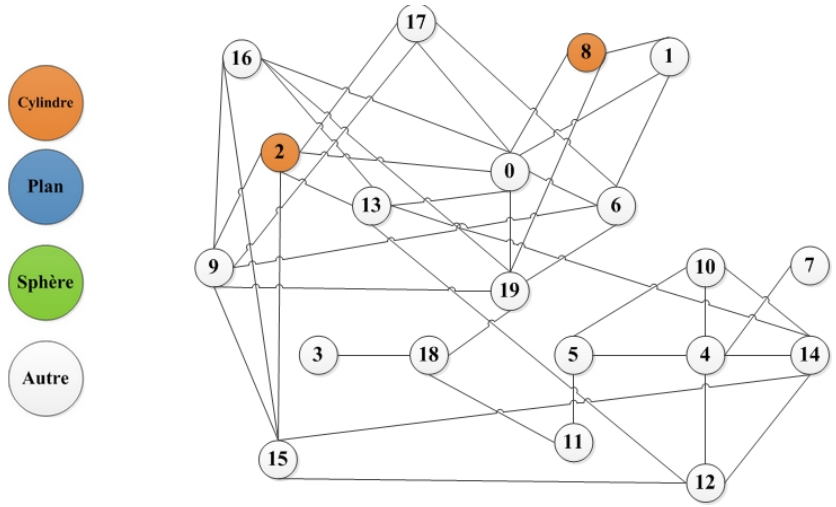


FIGURE 4.30 – Graphe de niveau 2 de l'assemblage scanné : les nœuds 2 et 8 sont deux cylindres.

Nous remarquons que seuls deux nœuds du graphe possèdent un attribut (de type cylindre).

Nous allons comparer le graphe ci-dessus avec les signatures de niveau 2 de la base de connaissances (étape n°5 de la figure 4.27). Pour chaque comparaison, nous avons limité le temps de calcul à 30 min.

Nous obtenons les scores ci-dessous :

Signature 1	**Signature 2**	**S 1**	**S 2**	**S 3**
signature_niv2_data_entree_20clusters	signature_niv2_bielle_assemblée_5clusters	20%	80%	32%
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_5clusters	25%	100%	40%

A partir des résultats obtenus ci-dessus, nous pouvons nous rendre compte qu'il est impossible de continuer notre scénario avec les étapes relatives à la signature de niveau 3. En effet, les signatures pré-sélectionnées lors de la comparaison de niveau 1 (étape n°3 du scénario) possèdent une signature de niveau 2 avec un nombre de clusters trop faible et les *mapping* issus de la comparaison de niveau 2 ne possèdent aucun nœud typé (ils sont tous de type "Autre"). Ainsi nous allons faire le choix d'"avorter" le scénario n°3 car les informations fournies jusqu'ici ne sont ni topologiques ni géométriques.

Nous allons poursuivre notre expérimentation à partir de l'étape n°4 du scénario n°5 et nous recommençons ainsi la comparaison de niveau 2. Nous obtenons les résultats présentés dans le tableau 4.21.

La figure 4.31 est un rappel du scénario n°5.

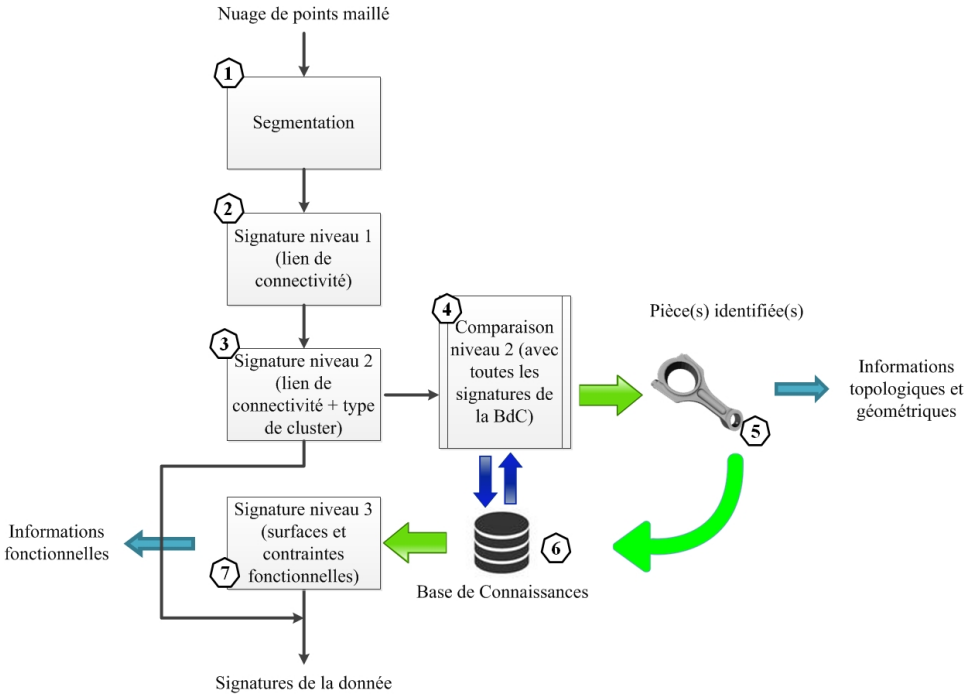


FIGURE 4.31 – Scénario n°5 pour la signature par graphe à trois niveaux.

Nous constatons que les deux signatures qui obtiennent le meilleur score sont celles avec le plus petit nombre de clusters (même allure que pour les résultats obtenus dans la section 4.3.2). La signature qui les précède avec 66% correspond à la "poulie_IFPEN_15clusters". Les signatures pour la bielle assemblée et le piston IFPEN avec un nombre de clusters compris entre 10 et 60 clusters (c'est-à-dire les signatures qui auraient dû, dans notre idéal, "matcher" avec un score élevé) obtiennent un score de similarité compris entre 8 et 60%, ce qui est faible par rapport aux meilleurs scores obtenus. Certaines comparaisons n'ont pu être achevées (notamment en ce qui concerne la signature de la "Bielle_Assemblée") car le calcul a été interrompu au bout de 30 min.

Afin de sélectionner les signatures qui serviront pour le niveau 3 (étape n°5 du scénario de la figure 4.31), deux possibilités s'offrent à l'utilisateur :

1. soit il garde les deux scores ayant obtenus 100 et 80% qui correspondent aux signatures respectives du "piston_ifpen" et de la "bielle_assemblée".
2. soit il choisit parmi les autres résultats de garder deux autres signatures ayant obtenu un score plus faible mais dont le nombre de clusters typés dans le *mapping* est plus important. Pour distinguer ces *mapping*, nous proposons de regarder manuellement chaque fichier .log et de chercher si les nœuds numéro 2 et 8 (d'après la figure 4.30) sont présents dans un des *mapping*. Une autre méthode vise à étudier tous les couples de nœuds des *mapping* obtenus et à regarder avec les signatures pour voir si les nœuds appartenant aux couples sont typés ou non ("Autre"). L'automatisation de cette tâche fait partie des perspectives.

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_data_entree_20clusters	signature_niv2_engrenage_helicoidal_15clusters	0	0	0
signature_niv2_data_entree_20clusters	signature_niv2_roue_dentee_1_15clusters	0	0	0
signature_niv2_data_entree_20clusters	signature_niv2_roue_dentee_1_20clusters	0	0	0
signature_niv2_data_entree_20clusters	signature_niv2_engrenage_helicoidal_25clusters	0	0	0
signature_niv2_data_entree_20clusters	signature_niv2_roue_dentee_1_25clusters	0	0	0
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_60clusters.xml	25	8	12
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_50clusters.xml	25	10	14
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_55clusters.xml	40	14	21
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_45clusters.xml	35	15	21
signature_niv2_data_entree_20clusters	signature_niv2_vilebrequin_2_25clusters	25	20	22
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_40clusters.xml	50	25	33
signature_niv2_data_entree_20clusters	signature_niv2_reducteur_4_20clusters	25	25	25
signature_niv2_data_entree_20clusters	signature_niv2_vilebrequin_2_20clusters	25	25	25
signature_niv2_data_entree_20clusters	signature_niv2_reducteur_4_25clusters	40	32	35
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_30clusters	55	36	44
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_10clusters	20	40	26
signature_niv2_data_entree_20clusters	signature_niv2_vilebrequin_2_15clusters	30	40	34
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_25clusters	55	44	48
signature_niv2_data_entree_20clusters	signature_niv2_poulie_ifpen_25clusters	55	44	48
signature_niv2_data_entree_20clusters	signature_niv2_bielle_assemblée_25clusters	60	48	53
signature_niv2_data_entree_20clusters	signature_niv2_poulie_ifpen_20clusters	50	50	50
signature_niv2_data_entree_20clusters	signature_niv2_reducteur_4_15clusters	40	53	45
signature_niv2_data_entree_20clusters	signature_niv2_bielle_assemblée_20clusters	55	55	55
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_20clusters	55	55	55
signature_niv2_data_entree_20clusters	signature_niv2_bielle_assemblée_10clusters	30	60	40
signature_niv2_data_entree_20clusters	signature_niv2_bielle_assemblée_15clusters	45	60	51
signature_niv2_data_entree_20clusters	signature_niv2_piston_ifpen_15clusters	45	60	51
signature_niv2_data_entree_20clusters	signature_niv2_bielle_5_15clusters	45	60	51
signature_niv2_data_entree_20clusters	signature_niv2_bielle_5_20clusters	60	60	60
signature_niv2_data_entree_20clusters	signature_niv2_poulie_ifpen_15clusters	50	66	57
signature_niv2_data_entree_20clusters	**signature_niv2_bielle_assemblée_5clusters**	20	**80**	32
signature_niv2_data_entree_20clusters	**signature_niv2_piston_ifpen_5clusters**	25	**100**	40

Tableau 4.21 – Résultats de la comparaison de niveau 2 avec la signature de l'assemblage scanné et l'ensemble de signatures en base.

Pour la première possibilité, nous avons étudié un des *mapping* obtenus pour chacune des signatures sus-citées et il en résulte les points suivants :

pour la bielle : le mapping avec la donnée d'entrée contient 4 nœuds dont 4 "Autre"

pour le piston : le mapping avec la donnée d'entrée contient 5 nœuds dont 5 "Autre".

Nous pouvons ainsi constater que les deux signatures présentant des similitudes avec notre donnée d'entrée ne fournissent aucune information topologique ou géométrique pour ce deuxième niveau de signature (alors que c'est le but de cette signature). Les signatures de niveau 2 de la base de connaissances ayant "matché" avec notre donnée d'entrée ne pourront être utilisées pour la signature de niveau 3 (pas de correspondance possible avec leur graphe de précédence respectif).

Pour la deuxième possibilité, nous avons donc étudié en détails les *mapping* de différents résultats de comparaison en prenant soin de noter le nombre de nœuds dans chaque *mapping* et de vérifier ensuite dans le fichier de signature si ces nœuds étaient typés ou non. Nous avons gardé uniquement les signatures dont le nombre de nœuds typés est d'au moins 1. Elles sont présentées ci-dessous :

Signature 2	**Score S2**	**Nb clusters mapping**	**Nb nœuds typés**
Bielle_assemblée_10clusters	66%	6	1 cylindre
Bielle_assemblée_25clusters	48%	12	1 cylindre
Piston_ifpen_50clusters	10%	5	1 cylindre
Piston_ifpen_55clusters	14%	8	1 cylindre
Piston_ifpen_60clusters	8%	5	1 cylindre
poulie_IFPEN_15clusters	66%	10	1 cylindre

Nous remarquons que la poulie obtient un score élevé par rapport à plusieurs composants que nous aurions aimé retrouver parmi les meilleurs scores. De plus, la similitude porte sur le même type de nœud (un cylindre). Nous pouvons ainsi nous rendre compte qu'il devient impossible de savoir quelle signature de la base de connaissances comporte des similitudes avec notre donnée d'entrée et ceci, malgré le fait que nous sélectionnons manuellement uniquement les signatures dont le *mapping* comporte au moins un nœud typé. La proposition faite en fin de section 4.3.3.2 d'ajouter une pondération ne paraîtrait pas suffisante. En effet, dans notre exemple la poulie ne serait pas écartée et obtiendrait probablement un des meilleurs scores de similarité.

Dans le cas de nos tests et en vue de passer au niveau 3, nous écartons ce mauvais résultat pour garder uniquement les signatures ayant eu un *mapping* avec au moins 1 nœud typé, tout en maximisant le nombre de nœuds dans le *mapping*, soit les signatures :

— la "Bielle_assemblée_25clusters"
— le "Piston_ifpen_50clusters".

Nous allons utiliser ces deux signatures pour la comparaison de niveau 3. La figure 4.32 illustre les deux sous-graphes communs pour chaque composant, obtenus suite à la comparaison avec notre donnée d'entrée.

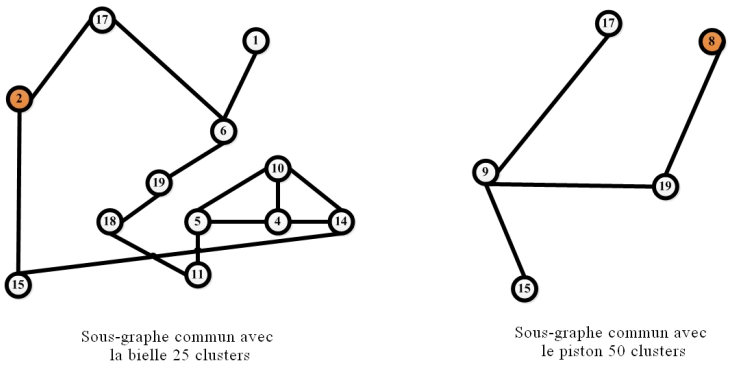


Figure 4.32 – Sous-graphes communs issus de la comparaison de niveau 2 entre la donnée d'entrée et une bielle et un piston de la base de connaissances.

La numérotation des nœuds des sous-graphes de la figure 4.32 correspond aux identifiants des nœuds du graphe de la donnée d'entrée, ceci dans le but de comparer les deux sous-graphes. Nous pouvons ainsi remarquer que certains nœuds du sous-graphe de gauche (*mapping* avec la bielle) sont également présents dans le sous-graphe de droite (*mapping* avec le piston). Il s'agit des nœuds numéro 15, 17 et 19. Nous pouvons en déduire que ces nœuds "en double" ne pourront être identifiés (étant donné qu'ils appartiennent à deux données différentes). De plus, étant de type "Autre", il sera impossible d'extraire leur géométrie. Dans le cadre de la signature de niveau 3, les numérotations des sous-graphes correspondront aux identifiants (numéros de nœud) des composants de la base de connaissances concernés.

4.4.4 La signature de niveau 3 et sa comparaison

La signature de niveau 3 (étape n°6 et 7 du scénario n°5) dépend complètement des clusters typés puisque le graphe de précédence s'appuie uniquement sur des surfaces fonctionnelles, c'est-à-dire des surfaces dont nous avons défini des caractéristiques (planéité, perpendicularité, coaxialité etc.).

Le changement de scénario en cours de route (voir section précédente) vise à maximiser le nombre de clusters typés dans le *mapping* et ainsi augmenter les chances d'identifier des surfaces fonctionnelles dans les données d'entrée. Cependant le nombre de nœuds typés de la donnée d'entrée (2 cylindres) ne permettra pas d'identifier plus de deux surfaces fonctionnelles. Ceci découle du choix du nombre de clusters initial pour la donnée d'entrée (choix dans Efpisoft : 20 clusters). Ce nombre ne pouvait être augmenté car il aurait rendu la comparaison de niveau 1 impossible car le temps de calcul aurait été supérieur à 5h, bien au delà des 30 minutes fixées dans ce test (cf. tableaux B.8, B.9 et B.10).

Nous partons ainsi des deux signatures citées précédemment ("Bielle_assemblée_25clusters" et "Piston_ifpen_50clusters") pour identifier les surfaces fonctionnelles dans notre donnée d'entrée puis générer sa signature de niveau 3. Nous allons alors récupérer les signatures de niveau 3 de ces deux signatures.

Nous commençons par traiter la signature pour la bielle. La figure 4.33 représente la mise en correspondance (*mapping*) de la signature de niveau 3 de la "Bielle_assemblée_25clusters" avec sa signature de niveau 2. Nous rappelons que cette étape est réalisée en amont de notre processus de RE grâce aux interfaces développées dans MATLAB et présentées en annexe C.1. Elle est cependant présentée à titre indicatif pour reprendre le déroulé des étapes telles que expliquées dans le chapitre Proposition.

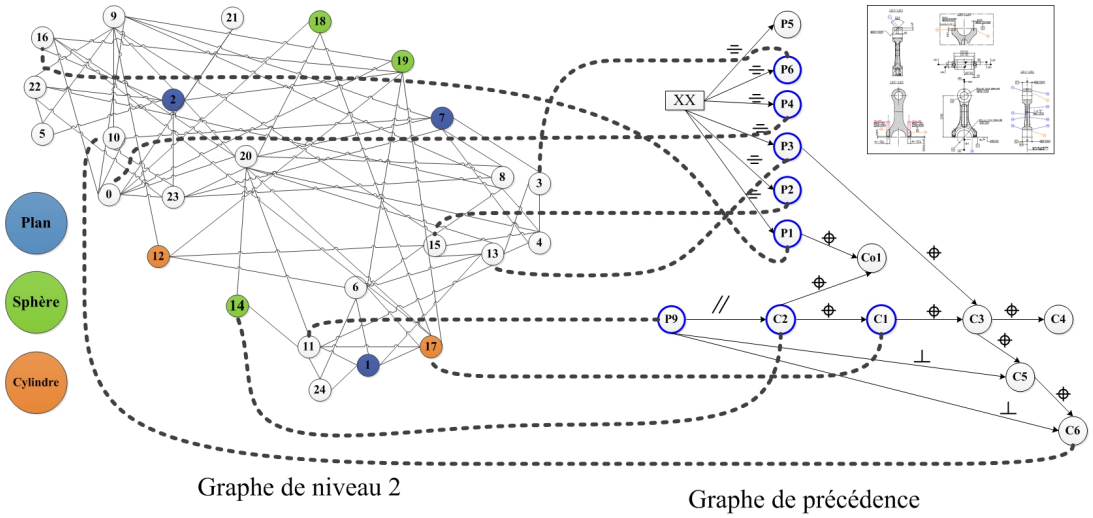


FIGURE 4.33 – Mise en correspondance du graphe de niveau 2 d'une bielle avec son graphe de niveau 3

Les nœuds du graphe de précédence (à droite) qui ont un nœud correspondant dans le graphe de niveau 2 (à gauche), ont été surlignés en bleu. Nous pouvons cependant observer que certains nœuds du graphe de précédence correspondent à des nœuds du graphe de niveau 2 de type "Autre".

La figure 4.34 correspond au *mapping* (fusion) entre les deux graphes de la figure 4.33. Le graphe obtenu correspond à $GPfus\_bielle$ tel que décrit dans la section 3.4.3 du chapitre Proposition.

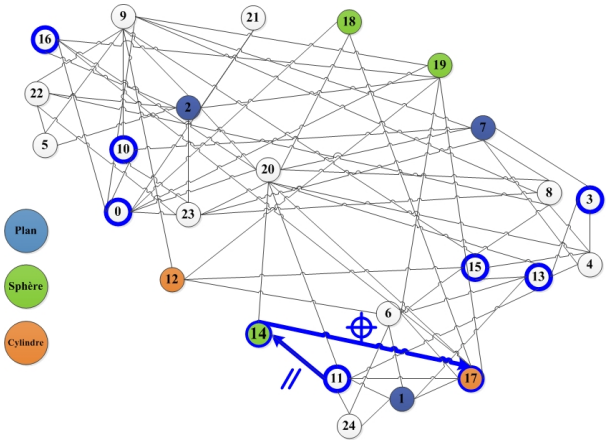


FIGURE 4.34 – *Mapping* entre les graphes de niveau 2 et 3 de la bielle en base.

A partir de ce niveau, nous allons comparer $GPfus\_bielle$ avec le sous-graphe obtenu lors de la comparaison de niveau 2 (cf. figure 4.32). Ce dernier, que nous nommerons SG_{2_bielle}, correspond à l'un des *mapping* obtenus lors de la comparaison entre le graphe de la donnée d'entrée et le graphe de la "Bielle_assemblée_25clusters". La figure 4.35 illustre cette étape de comparaison.

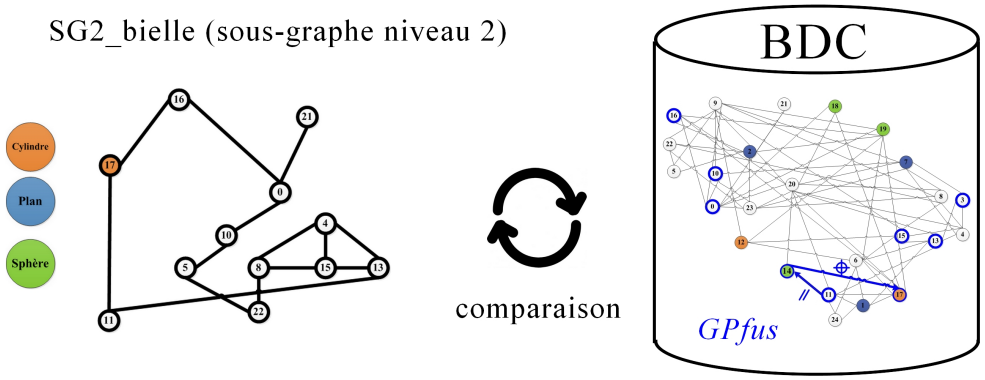


FIGURE 4.35 – Comparaison de niveau 3 entre $GPfus$ et le sous-graphe lié à la bielle.

Nous allons maintenant appliquer notre algorithme n°2 décrit dans le chapitre Proposition. La figure 4.36 représente les tableaux Tab_SG2_bielle et Tab_GPfus_bielle à l'initialisation de l'algorithme. Nous remarquerons qu'il existe seulement deux contraintes (car il y a seulement deux arcs dans $GPfus\_bielle$). Les autres nœuds "non liés" ont été ajoutés au tableau et leur existence sera tout de même vérifiée. Il y aura ainsi que deux itérations possibles (boucles du code).

Tab_SG2_bielle( )

Nœud 1
21
17
15
8
0
4
22
13
11
16
5
10

Tab_GPFus_bielle( )

Nœud 1	Nœud 2	Contrainte géométrique
11	14	Parallélisme
14	17	Localisation
0		
3		
10		
13		
15		
16		

Couple_Possible( )

Nœud 1	Nœud 2

Tab_GraphNiv3( )

Nœud 1	Nœud 2	Contrainte géométrique

FIGURE 4.36 – Initialisation des tableaux de l'algorithme de comparaison de graphe de niveau 3.

Nous commençons par parcourir le tableau Tab_GPFus_bielle :

- **1ère itération :** on compare la première valeur du Nœud 1 (11) avec les valeurs du tableau Tab_SG2_bielle. Ce dernier est parcouru ligne par ligne.
  - Si la valeur trouvée à la ligne de Tab_SG2 est égale à 11 alors on complète $Couple\_Possible(1,1)$ par la valeur 11. On compare maintenant la valeur de Nœud 2 de Tab_GPFus_bielle (14) avec les valeurs du tableau Tab_SG2_bielle.
    - Si la valeur trouvée à la ligne de Tab_SG2_bielle est égale à 14 alors on complète $Couple\_Possible(1,2)$ par la valeur 14.
    - Sinon on passe à la ligne suivante du tableau Tab_SG2_bielle jusqu'à ce qu'on ait parcouru toutes les lignes du tableau.
      Dans notre cas, la valeur 14 n'existe pas donc la première contrainte géométrique ne peut être vérifiée.
- **2ème itération :** de la même manière, on va chercher si les nœuds 14 et 17 du tableau Tab_GPFus_bielle existent dans Tab_SG2_bielle. Dans notre exemple, le nœud 14 n'existe pas dans Tab_SG2_bielle.
  Ce couple de nœuds ne peut pas alors être comptabilisé.

La figure 4.37 illustre les deux itérations décrites précédemment. Nous constatons ainsi qu'aucun couple de nœuds n'a pu être identifié soit aucune contrainte fonctionnelle ne peut être vérifiée.

Couple_Possible( )

	Nœud 1	Nœud 2
1ère itération	11	Ø
2e itération	Ø	17

Tab_GraphNiv3( )

Nœud 1	Nœud 2	Contrainte géométrique

FIGURE 4.37 – Illustration de l'algorithme de comparaison et affichage des couples possibles pour les deux itérations de l'algorithme.

A ce stade de la comparaison, nous avons identifié deux surfaces fonctionnelles (nœuds 11 et 17) en appliquant l'algorithme. Cependant il nous est impossible de vérifier les contraintes géométriques correspondantes. De plus, concernant le nœud n°11, même s'il appartient au $Tab\_GP_{Fus\_bielle}$ et à SG_{2_bielle}, il a été détecté comme "Autre" lors de l'étape de signature de niveau 2. On peut alors se poser la question sur son aptitude à être considéré comme une surface fonctionnelle.

Si nous regardons également les nœuds "non liés" de $GPfus\_bielle$ et que nous les comparons à ceux de SG_{2_bielle}, nous pouvons constater que les nœuds suivants existent dans le *mapping* : 0, 10, 13, 15 et 16. Pour le moment aucune suite n'a été donnée concernant l'existence de ces nœuds, mais nous pourrions imaginer extraire les informations géométriques de certaines d'entre eux et en particulier pour ceux qui sont typées. Ces informations permettraient le cas échéant de reconstruire la maquette numérique. Dans notre exemple, les nœuds 0, 10, 13, 15 et 16 sont de type "Autre" donc ceci n'est pas applicable et nous pouvons également nous poser la question sur leur aptitude à être considérés comme des surfaces fonctionnelles.

Ainsi par précaution et pour conclure sur la bielle, nous considèrerons uniquement le 17 comme surface fonctionnelle.

L'étape suivante consiste à réaliser la comparaison de niveau 3 pour le "Piston_ifpen_50clusters". Elle ne sera pas présentée dans ce manuscrit et nous exposerons uniquement les résultats dans la figure 4.38.

Tab_SG2_piston( )

Nœud 1
22
32
4
3
31

Tab_GPFus_piston( )

Nœud 1	Nœud 2	Contrainte géométrique
32	5	Coaxialité
5		
4		

Couple_Possible( )

	Nœud 1	Nœud 2
1ère itération	32	Ø

Tab_GraphNiv3( )

Nœud 1	Nœud 2	Contrainte géométrique

FIGURE 4.38 – Illustration de l'algorithme de comparaison sur le piston et affichage des couples possibles pour les deux itérations.

Nous pouvons remarquer qu'aucune contrainte fonctionnelle n'a pu être vérifiée, seul le nœud n°32 a été identifié comme une surface fonctionnelle. Nous pouvons remarquer également que le nœud n°4 fait partie des nœuds existants dans $Tab\_GP_{Fus\_piston}$. Cependant il a été détecté comme "Autre" donc, dans le doute, nous ne le considèrerons pas comme une surface fonctionnelle.

Pour conclure notre démarche, les surfaces fonctionnelles identifiées ne permettent pas de reconstruire un graphe car aucun arc (contrainte fonctionnelle) n'a pu être "généré". Seuls 2 nœuds (surfaces fonctionnelles) ont pu être identifiés dans la donnée d'entrée. Il s'agit des nœuds n°17 et 32. Si maintenant nous "traduisons" ces identifiants grâce aux deux *mapping* obtenus entre la donnée d'entrée et nos deux signatures en base (piston et bielle), nous pouvons déduire que les nœuds n°2 et 8 de la donnée d'entrée sont des surfaces fonctionnelles. Le tableau 4.22 illustre cette "traduction" en trois étapes pour la bielle.

Étape 1	Nœud identifié comme surface fonctionnelle pour la bielle	n°17
Étape 2	Mapping entre la bielle_25clusters et la donnée d'entrée (issu de la comparaison de niveau 2)	ID donnée entrée / ID bielle_25clusters 1 <-> 21 **2** <-> **17** 4 <-> 15 5 <-> 8 6 <-> 0 10 <-> 4 11 <-> 22 14 <-> 13 15 <-> 11 17 <-> 16 18 <-> 5 19 <-> 10
Étape 3	Nœud correspondant pour la donnée d'entrée	n°2

Tableau 4.22 – Traduction des identifiants de la bielle vers la donnée d'entrée en vue de retrouver les surfaces fonctionnelles correspondantes.

La figure 4.39 illustre la segmentation de la donnée d'entrée avec les clusters qui ont été identifiés comme des surfaces fonctionnelles (les clusters n°2 et 8 sont des cylindres).

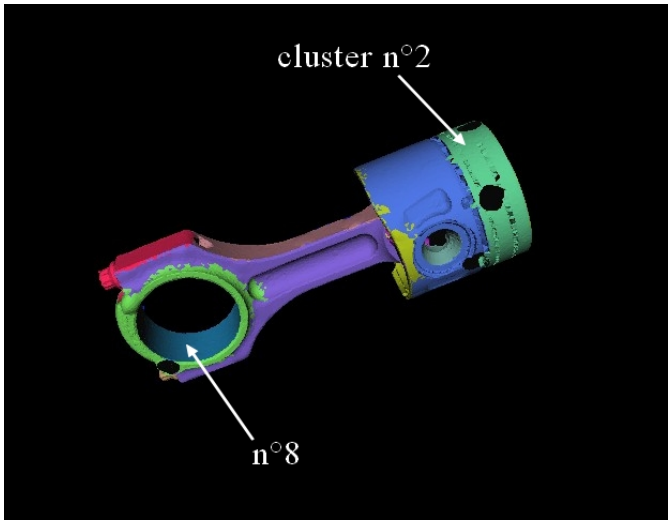


FIGURE 4.39 – Mise en évidence des clusters identifiés comme des surfaces fonctionnelles dans la donnée d'entrée.

L'expérimentation de notre méthodologie de signature par graphe sur un cas industriel a permis d'identifier seulement deux surfaces fonctionnelles. Le nombre restreint de clusters dans la donnée d'entrée est une des raisons principales. En effet, sa signature de niveau 2 contient seulement deux nœuds typés. Même si d'autres nœuds ont montré de possibles similitudes lors de l'étape de comparaison de niveau 3, nous sommes incapables de déduire qu'il s'agit également de surfaces fonctionnelles. Cette incertitude a été également mise en évidence lors de l'étape de comparaison de niveau 2 avec la poulie qui avait obtenu un score de similarité de 66% avec notre donnée d'entrée.

Nous avons pour cela effectué un test rapide en recommençant tout le scénario n°5 et en augmentant le nombre de clusters lors de l'étape de segmentation de la donnée d'entrée : nous l'avons segmentée en 30 clusters. La signature de niveau 2 de notre donnée d'entrée comporte alors 10 nœuds typés dont 6 sphères et 4 plans.

Les scores de similarité obtenus sont dans le tableau 4.23.

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	6	8	7
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	10	15	12
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_reducteur_4_25clusters.xml	13	16	14
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	16	20	18
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	23	28	25
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	20	30	24
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_reducteur_4_20clusters.xml	20	30	24
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	20	30	24
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	20	40	26
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_reducteur_4_15clusters.xml	20	40	26
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	33	50	40
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_bielle_5_15clusters.xml	26	53	35
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	26	53	35
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_piston_ifpen_20clusters.xml**	43	**65**	52
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_bielle_assemblée_15clusters.xml**	33	**66**	44
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_piston_ifpen_10clusters.xml**	23	**70**	35
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_bielle_assemblée_10clusters.xml**	23	**70**	35
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_piston_ifpen_15clusters.xml**	36	**73**	48
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_piston_ifpen_5clusters.xml**	13	**80**	22
signature_niv2_scan_bielle_piston_30clusters.xml	**signature_niv2_bielle_assemblée_5clusters.xml**	16	**100**	28

Tableau 4.23 – Résultats de la comparaison de niveau 2 avec la signature de l'assemblage scanné (30clusters) et l'ensemble des signatures en base.

Les signatures obtenant un score supérieur à 80% correspondent à celles avec 5 clusters pour la "bielle_assemblée" et le "piston_ifpen". Maintenant si nous observons les autres résultats obtenus avec un score de similarité compris entre 65 et 73%, nous remarquons qu'il s'agit des composants recherchés (c'est-à-dire la bielle assemblée et le piston IFPEN). Nous allons maintenant inspecter les *mapping* obtenus pour les composants suivants : le "piston_ifpen_20clusters" et la "bielle_assemblée_15clusters" qui ont obtenu respectivement un score de 65 et 66%. La figure 4.40 illustre deux des *mapping* extraits des fichiers .log de chaque comparaison.

Mapping avec piston_ifpen_20clusters	Type nœud
0 <-> 10	autre
1 <-> 16	autre
3 <-> 12	autre
5 <-> 2	autre
6 <-> 3	**sphère**
9 <-> 19	**sphère**
10 <-> 1	autre
11 <-> 14	autre
13 <-> 4	**sphère**
15 <-> 7	autre
18 <-> 9	autre
21 <-> 0	autre
23 <-> 15	autre

Mapping avec bielle_assemblée_15clusters	Type nœud
0 <-> 1	autre
3 <-> 14	autre
8 <-> 8	**sphère**
9 <-> 13	**sphère**
10 <-> 9	autre
14 <-> 0	autre
18 <-> 6	autre
20 <-> 2	autre
21 <-> 7	autre
24 <-> 5	autre

FIGURE 4.40 – Extraits des fichiers de *mapping* concernant les comparaisons et types de noeuds associés.

Nous constatons que le nombre total de clusters typés appartenant au deux *mapping* est bien supérieur que précédemment : nous avons 5 sphères. Ce sont donc 5 nœuds qui pourront être utilisés pour la comparaison de niveau 3 et également cinq surfaces fonctionnelles qui pourront être identifiées. Nous pouvons conclure de ce test que l'augmentation du nombre de clusters de la donnée d'entrée nous permet

d'obtenir des résultats de comparaison de niveau 2 de meilleure qualité. Nous pouvons ainsi croire au potentiel et à la faisabilité de la signature de niveau 3 moyennant une augmentation significative des capacités de calcul des machines réalisant les comparaisons.

Vers l'instanciation du composant

Dans cette section, nous allons mettre en lien notre méthodologie HDI-RE et l'étape de reconstruction de la maquette numérique et en particulier lorsqu'il s'agit d'une maquette comportant des modèles CAO paramétrés. Cette étape de notre processus n'a pas encore été implémentée et elle est présentée à titre prospectif.

Nous nous focaliserons sur l'étape de reconstruction du modèle CAO paramétré de la bielle que nous avons traitée dans la section précédente. Nous considérons l'existence dans notre base de connaissances d'un *template* CAO (cf. note 13 en bas de page) de cette bielle. Plusieurs méthodes existent pour créer et configurer ce *template*. Nous ferons l'hypothèse que c'est l'utilisateur qui s'en charge selon les pratiques de conception fixées dans son environnement de travail. Ainsi nous ne jugerons pas la pertinence des informations présentes dans cette donnée.

La figure 4.41 illustre une instance du *template* de la bielle dans l'environnement CATIA V5-6 R2013. Nous remarquons la présence de six paramètres dans l'arbre de construction. Dans notre exemple, il s'agit des paramètres pilotant du *template* CAO de la bielle. Nous nous intéresserons en particulier au paramètre "diamètre_1=24mm" car il correspond à celui d'un des cylindres de la bielle en base. Il se rapporte également à la surface fonctionnelle détectée dans la donnée d'entrée présentée dans la section précédente (cluster n°8 de la figure 4.39).

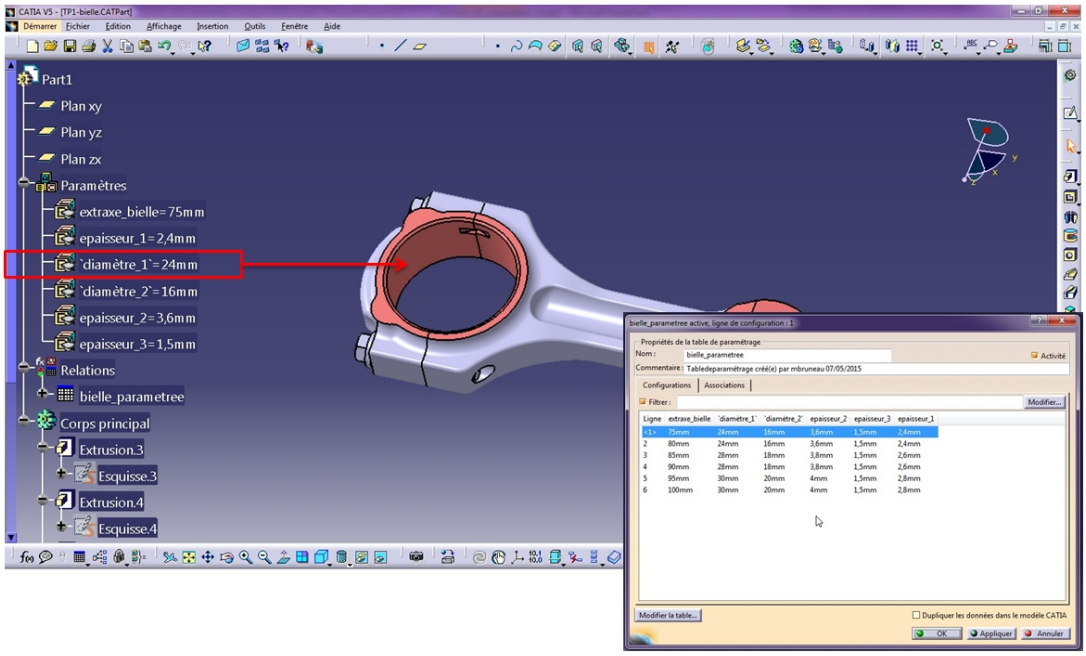


Figure 4.41 – Instanciation d'un *template* CAO d'une bielle paramétrée présente dans la base de connaissances dans CATIA V5-6 R2013.

Les paramètres de cette bielle sont liés à une table de paramétrage (fichier Excel^{TM}) que nous pouvons apercevoir en bas à droite de la figure 4.41. L'instance affichée correspond à la première ligne de la table de paramétrage. Cette ligne correspond à une configuration de la bielle, c'est-à-dire à une bielle dont les paramètres ont pris les valeurs de la ligne considérée.

Si nous revenons maintenant à notre donnée présentée dans la section précédente et que nous observons les caractéristiques géométriques du cluster n°8 (cf. figure 4.39), nous devons récupérer la valeur du diamètre du cylindre issu du maillage. Nous allons l'extraire des données récupérées lors de l'étape

de signature de niveau 2. Dans notre exemple, le diamètre de ce cylindre mesure 28 mm (valeur du *modèle_possible* tel que décrit dans l'algorithme 1).

Avec cette valeur, nous allons alors ouvrir la table de paramétrage et sélectionner la configuration pour laquelle le paramètre "diamètre_1" est égal à 28 mm. La figure 4.42 illustre la table de paramétrage issue du logiciel CATIA™.

	1	2	3	4	5	6	7
1	extraxe_bielle (mm)	`diamètre_1` (mm)	`diamètre_2` (mm)	epaisseur_2 (mm)	epaisseur_3 (mm)	epaisseur_1 (mm)	
2	75	24	16	3,6	1,5	2,4	
3	80	24	16	3,6	1,5	2,4	
4	85	28	18	3,8	1,5	2,6	
5	90	28	18	3,8	1,5	2,6	
6	95	30	20	4	1,5	2,8	
7	100	30	20	4	1,5	2,8	
8							
9							

Figure 4.42 – Table de paramétrage associée au *template* de la bielle en base

Nous remarquons que pour un "diamètre_1" égal à 28 mm, il s'agit de la configuration n°4. L'utilisateur peut ainsi générer automatiquement une bielle avec ce diamètre là et les valeurs des autres paramètres de la ligne du tableau.

Remarque : l'exemple présenté ci-dessus a volontairement été simplifié afin de présenter la démarche globale. Dans la réalité, un seul paramètre ne suffirait peut-être pas à instancier un *template* et générer le modèle CAO associé.

Enfin si nous imaginons reproduire ceci avec le piston et obtenir le modèle CAO équivalent, nous pourrons reconstruire la maquette numérique de l'assemblage bielle-piston avec les valeurs de paramètres issus de notre donnée scannée. La figure 4.43 représente l'affichage final de la maquette numérique.

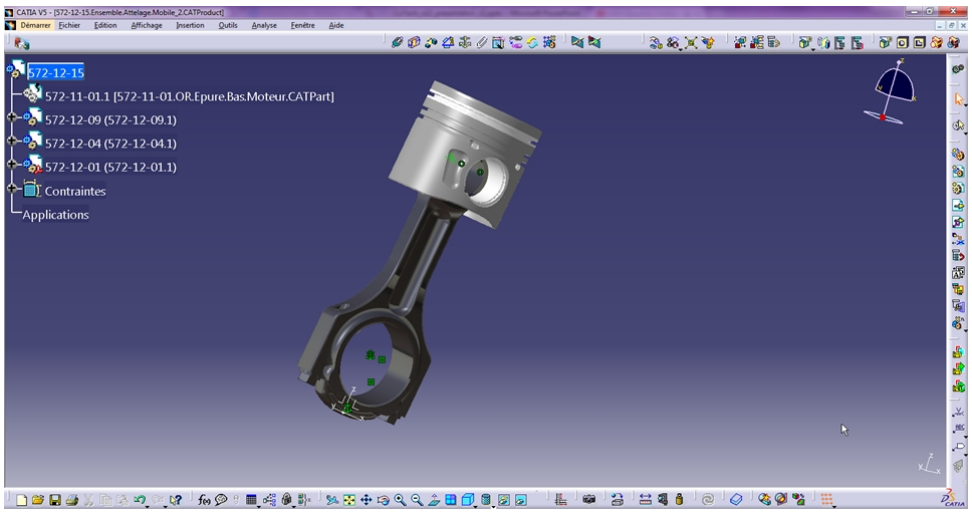


Figure 4.43 – Reconstruction de la maquette numérique de l'assemblage bielle-piston à l'aide des *templates* CAO instanciés dans CATIA V5-6 R2013.

Dans le futur, nous pourrons imaginer automatiser cette étape d'instanciation à partir de la signature de niveau 3 de notre donnée d'entrée.

4.4.5 Signature par critère iso-périmétrique

Cette section a pour but de présenter les résultats obtenus en appliquant notre signature par critère iso-périmétrique à notre donnée d'entrée.

Après application de l'algorithme de signature et d'après la formule 3.1 (section 3.3.2), nous obtenons un score de $C = 24.28$. En comparant cette valeur avec celles obtenues dans notre graphe qui regroupent toutes les valeurs obtenues (cf. figure 4.14), nous constatons que cette valeur croise les intervalles des familles d'assemblages suivantes : les moteurs et les réducteurs ; mais en aucun cas l'intervalle dédié à l'assemblage bielle-vilebrequin-piston (intervalle compris entre 7,57 et 17,09). Cependant si nous appliquons notre algorithme sur le même assemblage que celui de notre donnée d'entrée mais en utilisant des nuages de points issus de la CAO, nous obtenons $C = 23.83$, qui est une valeur très proche de celle obtenue précédemment.

Nous pouvons ainsi conclure que ce type de signature est pour le moment trop imprécis pour être utilisé pour reconnaître un assemblage. Seuls des composants et non des assemblages peuvent être identifiés les uns par rapport aux autres.

4.5 Conclusion sur l'ensemble des expérimentations réalisées

Nous pouvons déjà conclure que la présence de signatures de niveau 2 avec un nombre de nœuds typés inférieur à 1 dans la base de connaissances nous induit en erreur. En effet, les signatures ayant "matché" avec notre donnée d'entrée présentent certes des similitudes mais pas d'ordre topologique ou géométrique. De plus ces signatures ne peuvent être mises en relation avec leur signature de niveau 3 respectives car elles possèdent pour la plupart, un nombre de clusters trop faible.

Le nombre de clusters est un point crucial de ces travaux car leur quantité influe sur la qualité des signatures (niveau d'informations) et la capacité à la comparer (temps de calcul). De plus, le nombre de clusters typés dépend directement de l'algorithme de détection des surfaces canoniques. Il serait donc préférable de commencer par améliorer ce dernier grâce à l'expérimentation sur un panel plus large de données scannées. Puis après avoir re-généré les signatures de niveau 2 des composants de la base de connaissances, il faudrait alors étudier chaque signature et écarter celles qui possèdent moins de un cluster typé.

Enfin la génération des signatures de niveau 3 en amont du processus de RE à partir des signatures de niveau 2 pourrait s'avérer fastidieux. En effet, il nous faut réaliser la "conversion" niveau2-niveau 3 pour autant de signatures de niveau 2 qu'il en existe. Cependant dans certains cas (nombre de nœuds du graphe niveau 2 inférieur à 10), la signature de niveau 3 risque d'être impossible à générer (cluster correspondant à plusieurs surfaces fonctionnelles). A contrario si le nombre de nœuds des graphes de niveau 2 est supérieur au nombre de nœuds du graphe de précédence, alors plusieurs nœuds seront associés à la même surface fonctionnelle (nœud du graphe de précédence). Cependant cela ne poserait pas problème à notre méthodologie.

Conclusion générale et perspectives

" Choisissez un travail que vous aimez et vous n'aurez pas à travailler un seul jour de votre vie."

- Confucius

Ce manuscrit a présenté une méthodologie d'aide au Reverse Engineering (RE) d'assemblages mécaniques à partir de données hétérogènes nommée HDI-RE pour Heterogeneous Data Integration for Reverse Engineering. Cette méthodologie est dédiée à l'activité de Reverse Engineering et ceci dans un contexte routinier. En s'appuyant sur les concepts du Knowledge-Based Engineering (KBE) notamment en termes de capitalisation et de réutilisation des connaissances, HDI-RE a pour but de segmenter, signer des données hétérogènes puis de comparer ces signatures avec celles présentes dans une base de connaissances, en vue de calculer leur similarité. Au fur et à mesure de l'utilisation de la méthodologie, les connaissances sont stockées sous la forme de signatures. Ces dernières sont un ensemble de caractéristiques géométriques et topologiques permettant de décrire une donnée. Ces données peuvent être de différents types : en trois dimensions (nuages de points, modèles CAO, etc.), en deux dimensions (photographies, dessin de définition, etc.) ou zéro dimension (documents socio-techniques). Il existe alors au moins autant de mécanismes de signature (pour générer une signature) que de types de données.

En fonction du contexte de RE (*from scratch* / *as designed-as built* / *as designed-as maintained*) et du niveau de détails souhaité en sortie de processus de RE (maquette numérique avec modèles CAO paramétrés / nomenclature avec les enveloppes externes des modèles / nomenclature simple) un ou plusieurs mécanismes de signatures peuvent être employés.

Pour parvenir à extraire les informations nécessaires à la reconstruction de la maquette numérique ou de la nomenclature, HDI-RE fait appel à une solution logicielle pour la segmentation (Efpisoft). Pour la signature, des algorithmes ont été développés et testés dans ces travaux. Ils permettent de signer de deux manières : par critère global (avec un critère iso-périmétrique) et par graphe (graphe à trois niveaux). Enfin des algorithmes existants dans le domaine de la comparaison de graphes ont été adaptés à notre méthodologie puis ils ont été également testés. En fonction des scores de similarité obtenus avec les signatures de la base de connaissances, les composants présents dans les données d'entrée sont alors identifiés. Les connaissances extraites grâce au(x) mécanisme(s) de signature permettent ensuite de reconstruire la maquette numérique de l'assemblage.

Dans cette conclusion générale, nous proposons d'énumérer les caractéristiques de notre méthodologie HDI-RE puis nous annoncerons les perspectives.

Les caractéristiques de la méthode HDI-RE

La méthodologie HDI-RE (Heterogenous Data Integration for Reverse Engineering) propose d'adapter les méthodologies de reconnaissances de forme (*Shape Matching*) et de KBE (Knowledge-Based Engineering) à une problématique de RE. Les objectifs d'une telle méthodologie sont :

- de segmenter les données hétérogènes en entrée ;
- de sauvegarder les connaissances extraites des données segmentées ou non-segmentées sous la forme de signature ;
- de comparer les caractéristiques extraites (signatures) avec celles appartenant aux composants de la base de connaissances afin de reconnaître partiellement ou intégralement les pièces mécaniques présentes dans les données ;
- de réutiliser les connaissances enregistrées au cours des précédentes activités de RE afin d'identifier un ensemble de données ;
- d'accélérer le processus de RE en identifiant les surfaces fonctionnelles dans les données d'entrée en vue de reconstruire une maquette numérique avec des modèles CAO paramétrés.

Les différents algorithmes développés dans le cadre de ces travaux ont été intégrés à un démonstrateur nommé METIS, développé par la société DeltaCAD. Une brève présentation est faite en annexe C.2.

Quant à notre solution, elle s'applique dans les contextes suivants :

- **Mono-utilisateur :** les résultats (scores de similarité et caractéristiques géométriques et/ou fonctionnelles) en sortie de HDI-RE sont obtenus grâce à un seul utilisateur.
- **Mono-entreprise :** les cas d'utilisation de HDI-RE sont dédiés à une ou plusieurs familles de pièces ou assemblages propres à une entreprise (exemple de l'assemblage piston-bielle-vilebrequin).

La base de connaissances utilisée dans le cadre de ces travaux est composée des éléments suivants :

- de signatures de différents types (par critère global, par graphe) ;
- de signatures générées à partir de modèles "parfaits" (modèles complets issus de données CAO) et de signatures générées à partir de données incomplètes issues d'activités de RE antérieures ;
- de *templates* CAO permettant de reconstruire une maquette numérique à partir des caractéristiques extraites des données d'entrée.

La figure 4.44 résume les solutions apportées et présentées dans ce manuscrit.

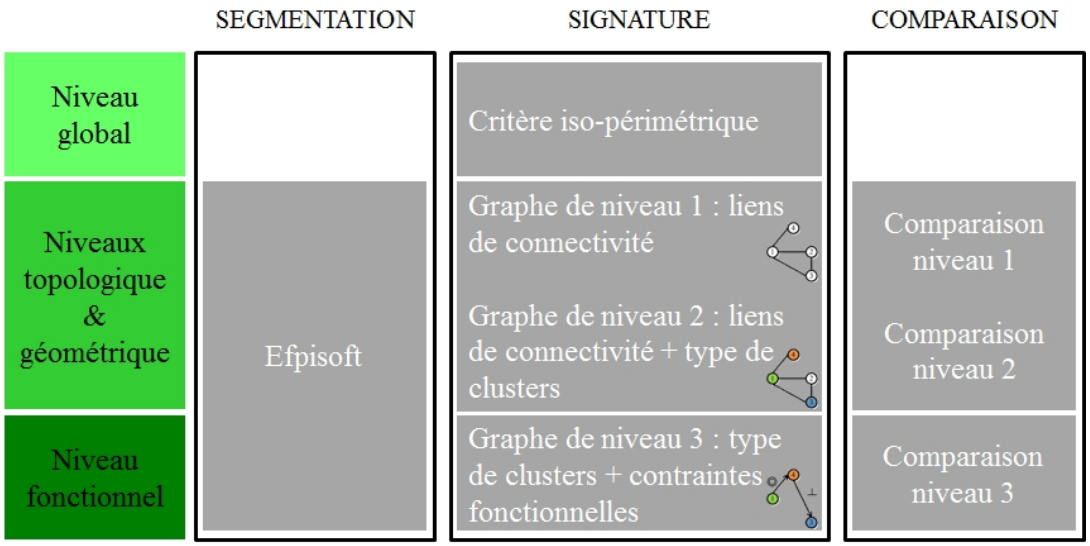


Figure 4.44 – Représentation globale des solutions apportées par HDI-RE.

En entrée de notre méthodologie, nous avons des données hétérogènes qui peuvent être :
— des données 3D : une maquette numérique, un maillage (nuage de points issu de scan) ;
— des données 2D : une photographie (couleur ou noir et blanc) ou un dessin de définition ;
— des données 0D (données socio-techniques) tels que des manuels de maintenance par exemple.

Nous retrouvons les trois principales étapes suivantes :
— **la segmentation :** elle permet de segmenter le nuage de points en régions de plus petite taille qui peuvent ensuite être ajustées à des surfaces canoniques. Cette étape est réalisée par le logiciel Efpisoft mais le nombre de régions est choisi par l'utilisateur.
— **la signature :** elle permet de décrire la donnée grâce aux caractéristiques extraites de la donnée elle-même ou de ses segments. Plusieurs types de signatures ont été développés : par critère iso-périmétrique ou par graphe à plusieurs niveaux. Le premier niveau de graphe correspond aux liens de connectivité entre les segments de la donnée. Le deuxième niveau de graphe est similaire au premier avec en plus le type de segment (cluster) qui est ajouté comme attribut à chaque nœud du graphe. Pour le troisième niveau, il s'agit des types de segments précédemment cités auxquels sont ajoutés des contraintes fonctionnelles après comparaison avec une signature de niveau 3 issue de la base de connaissances.
— **la comparaison des signatures :** elle est régie par un algorithme de comparaison de graphes pour ce qui concerne les graphes de niveaux 1 & 2 qui attribue un score de similarité entre les deux graphes (signatures de même type) comparés. Pour le niveau 3, il s'agit d'une comparaison de nœuds puis une vérification de contraintes géométriques (perpendicularité, coaxialité, parallélisme etc.) entre les nœuds préalablement identifiés.

Ce processus est itératif, les données sont segmentées, signées et comparées une à une jusqu'à ce que les informations nécessaires à la reconstruction de la maquette soient extraites.

Les données en sortie de HDI-RE sont de différents types :
— sous la forme d'un score de similarité : pour le critère iso-périmétrique, nous comparons la similarité de la valeur obtenue avec les autres valeurs de la base de connaissances. Pour les comparaisons de graphes de niveau 1 & 2, nous avons un pourcentage de similarité entre la signature de la donnée d'entrée et chaque signature considérée de la base de connaissances.
— sous la forme d'une description topologique et géométrique de la donnée d'entrée : les signatures par graphe de niveau 1 et 2 fournissent d'une part, des liens de connectivité entre les segments (faces) et d'autre part le type de segment (plan, cylindre ou sphère).
— sous la forme d'une description fonctionnelle : ceci concerne uniquement la signature de niveau 3 qui est basée sur les graphes de précédence utilisés en bureau des méthodes.

Selon le niveau de détails souhaité par l'utilisateur en sortie de son activité de RE (maquette numérique avec modèles CAO paramétrés / nomenclature avec enveloppes externes des modèles / nomenclature simple), plusieurs niveaux de signatures pourront être utilisés/combinés :
— le niveau global avec le critère iso-périmétrique permettra de reconstruire uniquement la nomenclature de la maquette numérique ;
— les niveaux topologique et géométrique avec les graphes de niveau 1 & 2 aideront à reconstruire une maquette avec des peaux et squelette (géométrie morte) ;
— le niveau fonctionnel avec le graphe de niveau 3 permettra de reconstruire une maquette avec des modèles CAO paramétrés.

La figure 4.45 illustre différents chemins possibles pour reconstruire une maquette numérique à partir de données hétérogènes et en fonction du niveau de détails souhaité en sortie. Les chemins en trait plein sont ceux qui découlent de nos travaux. Pour les traits en pointillés, il s'agit des chemins possibles, à étudier dans le futur (perspectives).

Remarque : les informations extraites lors de l'étape de signature de niveau 2 pourraient également être utilisées pour reconstruire une maquette avec des modèles CAO paramétrés, en considérant toutes les surfaces (nœuds du graphe). Le graphe de niveau 3 permet alors de réduire la quantité d'information à traiter puisque nous nous intéressons uniquement aux surfaces fonctionnelles.

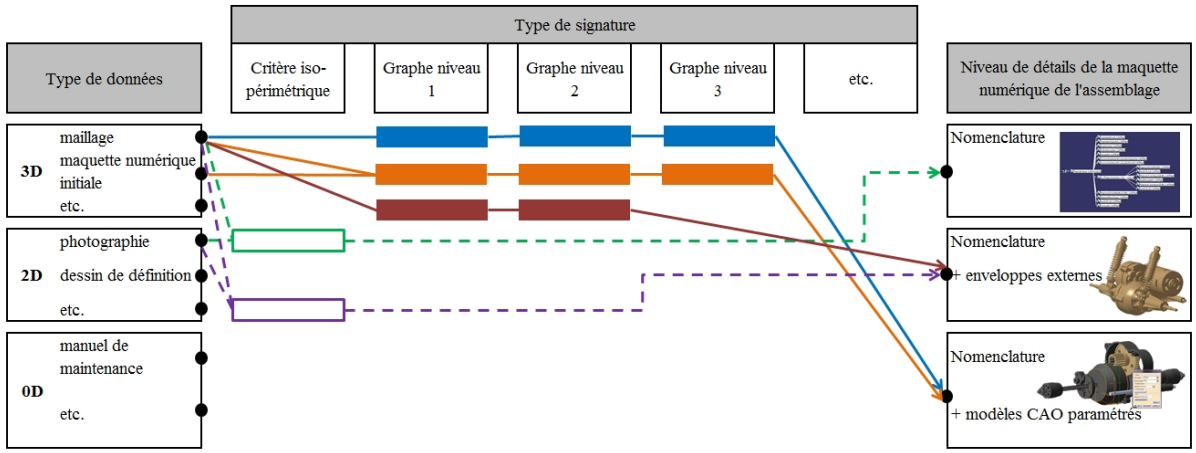


Figure 4.45 – Choix du type de signature afin de reconstruire la maquette numérique à partir de données hétérogènes en fonction du niveau de détails souhaité par l'utilisateur.

Critique de la méthode HDI-RE

La segmentation des données : le choix du nombre de clusters pour l'étape de comparaison a un impact sur l'étape de signature des données par graphe ainsi que sur leur comparaison. Tout d'abord, pour la signature, chaque cluster doit être ajusté à une surface canonique (plan, cylindre ou sphère). Cependant dans certains cas, aucun ajustement n'est possible et l'attribut "Autre" est ajouté au nœud du graphe considéré. Les expérimentations ont permis de mettre en exergue deux raisons principales : soit le cluster est trop "gros" et il pourrait être séparé en deux clusters de type différent (par exemple un plan et un cylindre). C'est notamment le cas lorsque le nombre de clusters de la donnée est faible (inférieur à 10). La deuxième raison est lorsque le cluster ne correspond à aucun type défini dans nos travaux comme par exemple pour les cônes ou les tores.

Ensuite le nombre de clusters impacte également l'étape de comparaison des signatures par graphe : chaque cluster correspondant à un nœud du graphe alors, plus le nombre de clusters est élevé, plus le graphe correspondant possède de nœuds. Suite aux expérimentations, nous pouvons déduire qu'à partir de 30 nœuds la comparaison de niveau 1 est impossible dans un laps de temps inférieur à cinq heures. Il en est de même pour les graphes de 50 clusters pour la comparaison de niveau 2.

Aucun test n'a permis de déterminer un nombre idéal de clusters et c'est à l'utilisateur de faire ce choix à partir de son expérience. Seule l'expérience de l'utilisateur permettra de choisir un nombre de clusters idéal.

Enfin même si les capacités de calcul venaient à être améliorées, il faudrait alors définir un maximum de clusters. Comme nous avons pu l'observer lors des tests, à partir d'une certaine valeur, la quantité de clusters typés stagne et le nombre de plans continue d'augmenter alors que les autres types (cylindre ou sphère) restent égaux à eux-même ou diminuent. D'après la technique employée dans Efpisoft, le nombre de clusters maximal est égal au nombre de triangles du maillage. D'où, plus le nombre de clusters tendra vers le nombre de triangles, plus les types de clusters détectés seront associés à des plans puisque qu'un triangle est ajusté à une surface plane lorsque nous nous plaçons dans un repère en trois dimensions.

La signature des données :

1. Pour la signature par critère iso-périmétrique : même si quelques familles de composants ont pu être identifiées grâce à des intervalles fixes de données, nous n'avons pas pu valider la robustesse de ce type de signature sur des composants incomplets (donnée sans surface fonctionnelle ou incomplète). Quant aux assemblages, la synthèse des résultats n'a pas permis de catégoriser les scores obtenus. Cette signature est d'autant plus globale que sa valeur est incertaine. Il nous parait donc impossible d'appliquer ce mécanisme de signature seul. Des expérimentations supplémentaires sur des assemblages plus proches (comparer uniquement des moteurs par exemple) permettrait peut-être de trouver un avantage à utiliser cette signature pour améliorer le processus de RE.

2. Pour la signature par graphe : pour ce qui concerne le niveau 1, nous pouvons déduire que ce type de graphe apporte un faible niveau d'information qui crée ensuite des confusions lors de l'étape de comparaison. Comme nous avons pu le remarquer, des signatures sont considérées comme similaires car leur graphe sont presque identiques. L'utilisation du scénario n°3 parait impossible. Pour le moment, seul le scénario n°5 permet de signer les données d'entrée jusqu'au niveau 3.

   Pour le niveau 2, des améliorations sont à apporter sur l'algorithme de détection. Les tests réalisés ont permis de fixer des valeurs pour les seuils. Cependant lorsque que nous avons pu re-tester l'algorithme sur des maillages un peu bruités (la bielle par exemple), il devient alors très difficile de typer les clusters à moins d'augmenter leur nombre (en tendant vers la valeur du nombre de triangles du maillage). D'autres tests mériteraient d'être réalisés sur des données moins bruitées que celles à disposition pour vérifier si les valeurs de seuil requièrent d'être augmentées ou non.

   Pour le niveau 3, nous avons pu constater que la génération de ce graphe est très limitée. En effet, beaucoup de paramètres jouent sur les possibilités d'atteindre ce niveau de signature pour la donnée d'entrée. Beaucoup d'efforts sont nécessaires pour sélectionner la signature de niveau 2 qui soit la plus apte pour permettre de signer fonctionnellement la donnée d'entrée. Le but initial de cette signature était d'éviter de comparer tous les nœuds du graphe afin de vérifier les possibles contraintes fonctionnelles. Cependant le résultat est qu'il ne reste plus suffisamment de nœuds en sortie du niveau 2 pour vérifier ces contraintes fonctionnelles. Les possibilités de régler ce problème sont limitées car deux facteurs sont à considérer : le nombre de clusters présents dans le *mapping* de niveau 2 (il faudrait d'abord réussir à obtenir des scores de similarité plus élevés) puis le nombre de clusters typés dans le *mapping* cité précédemment.

La comparaison des signatures : le nombre de clusters est le point bloquant de cette étape. Comme nous l'avons constaté, les comparaisons nécessitent de plus en plus de temps lorsque le nombre de clusters augmente notamment en ce qui concerne le niveau 1 ; le nombre maximal possible étant d'à peine une trentaine de nœuds.

Ensuite concernant le niveau 2, les scores de similarité ne prennent pas en compte le nombre de clusters typés. Deux solutions seraient envisageables : la première serait de ne pas ajouter d'attribut aux nœuds de type "Autre", ce qui reviendrait dans certains cas à de la comparaison de niveau 1 et donc limiterait de nouveau le nombre de clusters. Une deuxième solution serait de considérer le nombre de clusters typés dans le *mapping* dans le calcul du score de similarité.

Enfin concernant le niveau 3, l'algorithme de comparaison n'ayant pas été testé sur plusieurs données (seulement testé "à la main"), il est difficile d'évaluer sa robustesse. De nombreux tests sont à réaliser également concernant les solutions choisies pour vérifier les contraintes fonctionnelles (perpendicularité, coaxialité, etc.).

Les perspectives

Ces travaux ont été financés dans le cadre du projet ANR METIS dont l'échéance était au 30 septembre 2015. De nombreuses perspectives sont à envisager à court et moyen termes concernant les résultats présentés dans ce manuscrit.

Les perspectives à court terme :

Concernant l'étape de **segmentation**, des recherches sont en cours afin de parvenir à extraire les types de clusters du logiciel Efpisoft. Cette information est disponible au niveau interne du logiciel puisque les calculs effectués pour la segmentation reposent sur la capacité du cluster considéré à s'ajuster au mieux à une surface canonique donnée. D'autres tests sont à réaliser afin de définir un nombre idéal de clusters pour segmenter une donnée. Comme nous avons pu le remarquer, les améliorations apportées sur l'algorithme de Fitting (cf. conclusions de la section 4.2.1) affectent le nombre total de clusters de la donnée nécessaires pour obtenir une signature de niveau 2 avec un nombre de clusters typés suffisant.

En ce qui concerne l'étape de **signature**, l'algorithme *FittingCylinder* pour le niveau 2 mériterait d'être amélioré afin de réduire la valeur du seuil. Comme cité précédemment, d'autres tests seront menés sur d'autres données scannées afin de rectifier si nécessaire les trois valeurs de seuil de détection de surface canonique.

Enfin pour l'étape de **comparaison de signatures**, il est envisagé d'aider l'utilisateur dans le choix de la meilleure signature (en sortie de la comparaison de niveau 2). Pour cela, un algorithme de traitement des résultats pourrait être développé. Il permettrait de mettre en évidence les signatures ayant obtenu des *mapping* avec au moins un nœud typé ; tout ceci dans le but de signer la donnée avec le niveau 3. Enfin, le codage de l'algorithme de signature de niveau 3 est à effectuer, que ce soit concernant la vérification des nœuds mais aussi la vérification des contraintes fonctionnelles. Des tests seront également à prévoir pour tester ensuite la robustesse de ces algorithmes.

Les perspectives à moyen terme :

L'aspect "données hétérogènes" n'ayant pu être développé suffisamment dans le cadre de ces travaux, d'autres algorithmes de segmentation, de signature et de comparaison pourront être envisagés et testés. En effet, tel que nous l'avions relevé dans les travaux de [Deveau, 2006], les images pourraient nous aider à déterminer les limites entre les composants d'un assemblage scanné. Elles permettraient également de fournir d'autres informations tel que le matériau par exemple. L'étude de documents texte reste également à définir.

De plus, il serait pertinent de tester l'ensemble de la méthodologie HDI-RE dans un contexte de données massives et avec une base de connaissances propre à une entreprise. Les travaux dans les domaines du *Deep Learning*[3] ou du *Data Mining* seraient des pistes intéressantes de recherche. Enfin des tests sur des assemblages de plus grande taille (nombre de pièces supérieur à 10) permettraient également de tester la robustesse de la méthodologie dans sa globalité.

Les publications scientifiques

Ces travaux de thèse ont fait l'objet de publications scientifiques à trois conférences internationales avec comité de lecture :

- Ingegraph [Bruneau *et al.*, 2013] : "Towards new processes to reverse engineering digital mock-ups from a set of heterogeneous data", Proceedings of the Ingegraph- ADM- AIP Primeca Conference, 19-21 Juin 2013, Madrid, Espagne.
- TMCE 2014 [Bruneau *et al.*, 2014] : "A methodology of reverse engineering for large assemblies products from heterogeneous data", Proceedings of the TMCE 2014, 19-23 Mai 2014, Budapest, Hongrie.
- TMCE 2016 [Bruneau *et al.*, 2016] : "A three-levels signature by graph for Reverse Engineering of mechanical assemblies", 9-13 Mai 2016, Aix-en-Provence, France.

3. "Le *Deep learning* est un ensemble d'algorithmes utilisés dans le domaine du *machine learning* (apprentissage automatique), afin de modéliser avec un haut niveau d'abstraction des données grâce à des architectures qui sont composées de différentes transformations non linéaires" d'après [Technopedia, 2015].

Annexes

ANNEXE A

Codes et algorithmes

A.1 Code XML de conversion de la segmentation

Les trois pages suivantes sont un extrait du fichier XML qui est la conversion de la scène Open Inventor issue du logiciel Efpisoft. Le nombre de pages a été volontairement réduit pour plus de lisibilité et par conséquent la liste de *vertex* a été tronquée. L'architecture du fichier est la suivante :

Nom du fichier

- *VextexList* : liste de tous les sommets (points du nuage) numérotés de 0 à 4258
  - x : coordonnée suivant x du point
  - y : coordonnée suivant y du point
  - z : coordonnée suivant z du point
- *RegionList* : liste des régions (clusters) possédant chacune un attribut correspondant à une couleur (code propre à Open Inventor)
  - facet : liste des triangles appartenant à chaque région
    - vertexNum1 : numéro du premier sommet du triangle
    - vertexNum2 : numéro du deuxième sommet
    - vertexNum3 : numéro du troisième sommet

La segmentation étant de 12 clusters, nous retrouvons alors 12 régions.

```
<?xml version="1.0" encoding="UTF-8"?>
<ivFileData>
  - <VertexList>
    - <vertex number="0">
        <x>-55.54912186</x>
        <y>26.48835754</y>
        <z>-10.00000000</z>
      </vertex>
    - <vertex number="1">
        <x>-57.15165329</x>
        <y>26.32280540</y>
        <z>-10.00000000</z>
      </vertex>
    + <vertex number="2">
    + <vertex number="3">
    + <vertex number="4">
    + <vertex number="5">
    + <vertex number="6">
    + <vertex number="7">
    + <vertex number="8">
    + <vertex number="9">
    + <vertex number="10">
    + <vertex number="11">
    + <vertex number="12">
    + <vertex number="13">
    + <vertex number="14">
    + <vertex number="15">
    + <vertex number="16">
    + <vertex number="17">
    + <vertex number="18">
    + <vertex number="19">
    + <vertex number="20">
    + <vertex number="21">
    + <vertex number="22">
    + <vertex number="23">
    + <vertex number="24">
    + <vertex number="25">
    + <vertex number="26">
    + <vertex number="27">
    + <vertex number="28">
    + <vertex number="29">
    + <vertex number="30">
    + <vertex number="31">
    + <vertex number="32">
    + <vertex number="33">
    + <vertex number="34">
    + <vertex number="35">
    + <vertex number="36">
    + <vertex number="37">
    + <vertex number="38">
    + <vertex number="39">
    + <vertex number="40">
    + <vertex number="41">
```

```
    + <vertex number="4238">
    + <vertex number="4239">
    + <vertex number="4240">
    + <vertex number="4241">
    + <vertex number="4242">
    + <vertex number="4243">
    + <vertex number="4244">
    + <vertex number="4245">
    + <vertex number="4246">
    + <vertex number="4247">
    + <vertex number="4248">
    + <vertex number="4249">
    + <vertex number="4250">
    + <vertex number="4251">
    + <vertex number="4252">
    + <vertex number="4253">
    + <vertex number="4254">
    + <vertex number="4255">
    + <vertex number="4256">
    + <vertex number="4257">
    + <vertex number="4258">
  </VertexList>
- <RegionList>
  - <region color="0xbc09acff">
    - <facet>
        <vertexNum1>1006</vertexNum1>
        <vertexNum2>164</vertexNum2>
        <vertexNum3>1005</vertexNum3>
      </facet>
    - <facet>
        <vertexNum1>342</vertexNum1>
        <vertexNum2>20</vertexNum2>
        <vertexNum3>1008</vertexNum3>
      </facet>
    - <facet>
        <vertexNum1>1003</vertexNum1>
        <vertexNum2>341</vertexNum2>
        <vertexNum3>896</vertexNum3>
      </facet>
    - <facet>
        <vertexNum1>1000</vertexNum1>
        <vertexNum2>339</vertexNum2>
        <vertexNum3>337</vertexNum3>
      </facet>
    - <facet>
        <vertexNum1>336</vertexNum1>
        <vertexNum2>338</vertexNum2>
        <vertexNum3>1002</vertexNum3>
      </facet>
    - <facet>
        <vertexNum1>996</vertexNum1>
        <vertexNum2>21</vertexNum2>
        <vertexNum3>998</vertexNum3>
```

```
                    <vertexNum3>695</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>885</vertexNum1>
                    <vertexNum2>693</vertexNum2>
                    <vertexNum3>1040</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>344</vertexNum1>
                    <vertexNum2>1040</vertexNum2>
                    <vertexNum3>693</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>885</vertexNum1>
                    <vertexNum2>695</vertexNum2>
                    <vertexNum3>693</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>697</vertexNum1>
                    <vertexNum2>696</vertexNum2>
                    <vertexNum3>698</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>971</vertexNum1>
                    <vertexNum2>698</vertexNum2>
                    <vertexNum3>694</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>694</vertexNum1>
                    <vertexNum2>698</vertexNum2>
                    <vertexNum3>696</vertexNum3>
                </facet>
            - <facet>
                    <vertexNum1>460</vertexNum1>
                    <vertexNum2>522</vertexNum2>
                    <vertexNum3>808</vertexNum3>
                </facet>
        </region>
      + <region color="0x36f628ff">
      + <region color="0x3582d5ff">
      + <region color="0xa9a068ff">
      + <region color="0xcf8e2cff">
      + <region color="0x57ec2bff">
      + <region color="0xfe081eff">
      + <region color="0x90ad77ff">
      + <region color="0x4a57e4ff">
      + <region color="0x9dc908ff">
      + <region color="0xf41d46ff">
      + <region color="0xaaac51ff">
    </RegionList>
</ivFileData>
```

A.2 Algorithme de signature de niveau 1

Les informations concernant la description de cet algorithme ont été fournies par Laurent Vallet, un des collaborateurs sur le projet METIS.

— Chargement d'un fichier ".iv" et construction des polyèdres.

$$:: loadIvFile(const char * filename, bool signEnable)$$

Les étapes sont les suivantes :

1. Lecture du fichier ".iv" avec les informations : vertex, orderedRGBA, coordIndex, materialIndex.
2. Instanciation et remplissage des tableaux : points, facets, regions, mapping facet-region.
3. Création des polyèdres relatifs aux régions.
4. Signature niveau 1 sur les polyèdres (le polyèdre global + les polyèdres relatifs à chacune des régions). Par exemple pour un polyèdre p1 :

$$p1 \rightarrow computeMinSphere()$$
$$p1 \rightarrow computeCenterOfMass()$$
$$p1 \rightarrow computeBoundingBox()$$
$$p1 \rightarrow computeSumOfFacetAreas()$$
$$p1 \rightarrow computeIsoperimetricCriterion()$$
$$p1 \rightarrow computeConvexHull()$$
$$p1 \rightarrow computeConvexHullArea()$$
$$p1 \rightarrow computeConvexHullIsoperimetricCriterion()$$

5. Calcul des liens de connexion entre les *region groups*. On compare chaque facette d'une première région avec chaque facette d'une seconde région.
6. Export de la signature dans un fichier XML :

$$t.generateXMLDocument(filename, vectorOfSegments, loadedPolyhedron, pairSetConnectedSegments)$$

— Exportation de chaque polyèdre (cluster) dans un fichier OFF.

$$void Scene :: exportAllPolyedronsToOffFiles(const char * filename)$$

Les fichiers OFF sont créés dans le répertoire "/region_group/".

A.3 Code XML de la signature de niveau 1 d'une bielle

Ce fichier XML (page suivante) correspond au graphe de signature de premier niveau de la bielle illustrée ci-dessous. Le détail de l'architecture du fichier est décrit à la section 3.3.1.1.

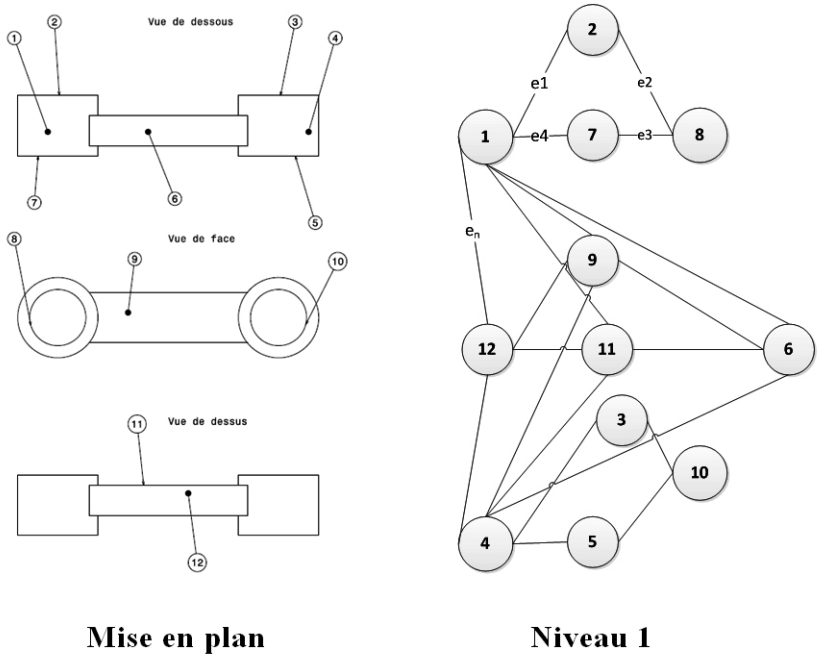


Figure A.1 – Une bielle et son graphe de niveau 1.

<?xml version="1.0" encoding="UTF-8"?>
<Signature type="**Connectivity graph**">
<!--Dates are formatted mm/dd/yy. Don't forget XML is case-sensitive.-->
- <MainPolyhedron>
- <Center type="**OfMasse**">
<x>0.00142177</x>
<y>12.60532996</y>
<z>6.07104017</z>
</Center>
- <ApproximationPrimitive type="**BoundingBox**">
<volume>179955.00000000</volume>
<xMin>-75.00000000</xMin>
<xMax>75.00000000</xMax>
<yMin>-7.49437189</yMin>
<yMax>32.49437332</yMax>
<zMin>-10.00000000</zMin>
<zMax>20.00000000</zMax>
</ApproximationPrimitive>
- <FacetAreas>
<sum>20567.44000000</sum>
</FacetAreas>
- <IsoperimetricCriterion>
<value>16.39000000</value>
</IsoperimetricCriterion>
</MainPolyhedron>
- <RegionList>
- <Region number="**0**" color="**#3582d5**">
+ <Center type="**OfMasse**">
+ <ApproximationPrimitive type="**BoundingBox**">
+ <FacetAreas>
+ <IsoperimetricCriterion>
</Region>
+ <Region number="**1**" color="**#36f628**">
+ <Region number="**2**" color="**#4a57e4**">
+ <Region number="**3**" color="**#57ec2b**">
+ <Region number="**4**" color="**#90ad77**">
+ <Region number="**5**" color="**#9dc908**">
+ <Region number="**6**" color="**#a9a068**">
+ <Region number="**7**" color="**#aaac51**">
+ <Region number="**8**" color="**#bc09ac**">
+ <Region number="**9**" color="**#cf8e2c**">
+ <Region number="**10**" color="**#f41d46**">
+ <Region number="**11**" color="**#fe081e**">
</RegionList>
- <Connectivity>
<Link regionNum2="**1**" regionNum1="**0**"/>
<Link regionNum2="**3**" regionNum1="**0**"/>
<Link regionNum2="**4**" regionNum1="**0**"/>
<Link regionNum2="**6**" regionNum1="**0**"/>
<Link regionNum2="**9**" regionNum1="**0**"/>
<Link regionNum2="**11**" regionNum1="**0**"/>
<Link regionNum2="**8**" regionNum1="**1**"/>
<Link regionNum2="**3**" regionNum1="**2**"/>

```
            <Link regionNum2="4" regionNum1="2"/>
            <Link regionNum2="5" regionNum1="2"/>
            <Link regionNum2="9" regionNum1="2"/>
            <Link regionNum2="10" regionNum1="2"/>
            <Link regionNum2="11" regionNum1="2"/>
            <Link regionNum2="9" regionNum1="3"/>
            <Link regionNum2="11" regionNum1="3"/>
            <Link regionNum2="9" regionNum1="4"/>
            <Link regionNum2="11" regionNum1="4"/>
            <Link regionNum2="7" regionNum1="5"/>
            <Link regionNum2="8" regionNum1="6"/>
            <Link regionNum2="10" regionNum1="7"/>
        </Connectivity>
</Signature>
```

A.4 Résultat de comparaison de niveau 1 entre deux graphes

Le fichier texte sur les deux pages suivantes est un extrait du fichier de résultats issu de la comparaison de graphe de niveau 1 à l'aide de l'algorithme *mcgregor_common_subgraphs* appartenant à la bibliothèque Boost Graph.

Le fichier est structuré ainsi :

— affichage de la liste des nœuds du premier graphe : "0 $<-->$ 4 9 15 17 28 32 37" signifie que le nœud d'indice 0 de la signature nommée "test_vilebrequin.xml" est relié aux nœuds suivants : 4, 9, 15, 17, 28, 32 et 37 ;

— affichage de la liste des nœuds du deuxième graphe nommé "signature_vilebrequin_bdd_10clusters" ;

— affichage de la liste de tous sous-graphes communs aux deux graphes (mapping du graphe). Dans notre exemple, il en existe plus d'une centaine. La liste a été réduite à quelques uns. Pour chaque ligne du mapping, chaque nœud du premier graphe est associé au nœud équivalent dans le deuxième graphe pour le sous-graphe considéré.

— les valeurs de similarité entre les deux graphes (les explications sont données dans la section 3.4.1).

```
First graph (Test_3/test_vilebrequin.xml) :
0 <--> 4 9 15 17 28 32 37
1 <--> 34 37
2 <--> 7 10 14 19 31
3 <--> 8
4 <--> 0
5 <--> 14 38
6 <--> 11 18
7 <--> 2 10 19 20 25 27 39
8 <--> 3 30 33
9 <--> 0
10 <--> 2 7 25 31
11 <--> 6 38
12 <--> 15 22
13 <--> 17 26 36 37
14 <--> 2 5 19 31 34 35
15 <--> 0 12 17 26 28
16 <--> 22 23
17 <--> 0 13 15 37
18 <--> 6 21
19 <--> 2 7 14 25 35
20 <--> 7
21 <--> 18
22 <--> 12 16
23 <--> 16 30
24 <--> 26 28 36 37
25 <--> 7 10 19 31 34 35
26 <--> 13 15 24 28 36
27 <--> 7
28 <--> 0 15 24 26 37
29 <--> 33
30 <--> 8 23
31 <--> 2 10 14 25 34
32 <--> 0
33 <--> 8 29
34 <--> 1 14 25 31 35
35 <--> 14 19 25 34
36 <--> 13 24 26 37
37 <--> 0 1 13 17 24 28 36
38 <--> 5 11
39 <--> 7
Second graph (signature_vilebrequin_bdd_10clusters.xml) :
0 <--> 4
1 <--> 4 5 6
2 <--> 6 7 8
3 <--> 6 8
4 <--> 0 1 5
5 <--> 1 4 7
6 <--> 1 2 3
7 <--> 2 5
8 <--> 2 3 9
9 <--> 8

mcgregor_common_subgraphs_maximum_unique mapping :
0 <-> 1
1 <-> 7
2 <-> 3
5 <-> 9
14 <-> 8
26 <-> 0
28 <-> 4
34 <-> 2
37 <-> 5
---
0 <-> 1
1 <-> 7
2 <-> 9
5 <-> 3
14 <-> 8
26 <-> 0
```

```
28 <-> 4
34 <-> 2
37 <-> 5
---
0 <-> 1
1 <-> 7
5 <-> 3
14 <-> 8
19 <-> 9
26 <-> 0
28 <-> 4
34 <-> 2
37 <-> 5
---
0 <-> 1
1 <-> 7
5 <-> 9
14 <-> 8
19 <-> 3
26 <-> 0
28 <-> 4
34 <-> 2
37 <-> 5
---
0 <-> 1
1 <-> 7
10 <-> 3
19 <-> 9
25 <-> 8
26 <-> 0
28 <-> 4
34 <-> 2
37 <-> 5
---
1 <-> 9
5 <-> 7
6 <-> 0
11 <-> 4
14 <-> 2
19 <-> 6
25 <-> 3
34 <-> 8
38 <-> 5
---
12 <-> 7
13 <-> 3
15 <-> 2
16 <-> 4
17 <-> 6
22 <-> 5
23 <-> 0
24 <-> 9
26 <-> 8
---
Similary algorithm kind : average of difference of graph vertex number
Similarity 1 : V_subG * 100 / V_G1
Similarity 2 : V_subG * 100 / V_G2
Similarity 3 : 2 * V_subG * 100 / (V_G1 + V_G2)

1 - Graph1 and subgraph similarity is : 22%
2 - Graph2 and subgraph similarity is : 90%
3 - Global similarity is : 36%
```

ANNEXE B

Tableaux

B.1 Résultats des tests de comparaison de niveau 1 pour la bielle et le vilebrequin

Les résultats présentés dans le tableau B.1 concernent les tests sur une bielle scannée pour la comparaison de niveau 1. Pour chaque test, la signature de la donnée scannée est comparée à un ensemble de données dont la description est faite dans la figure B.1.

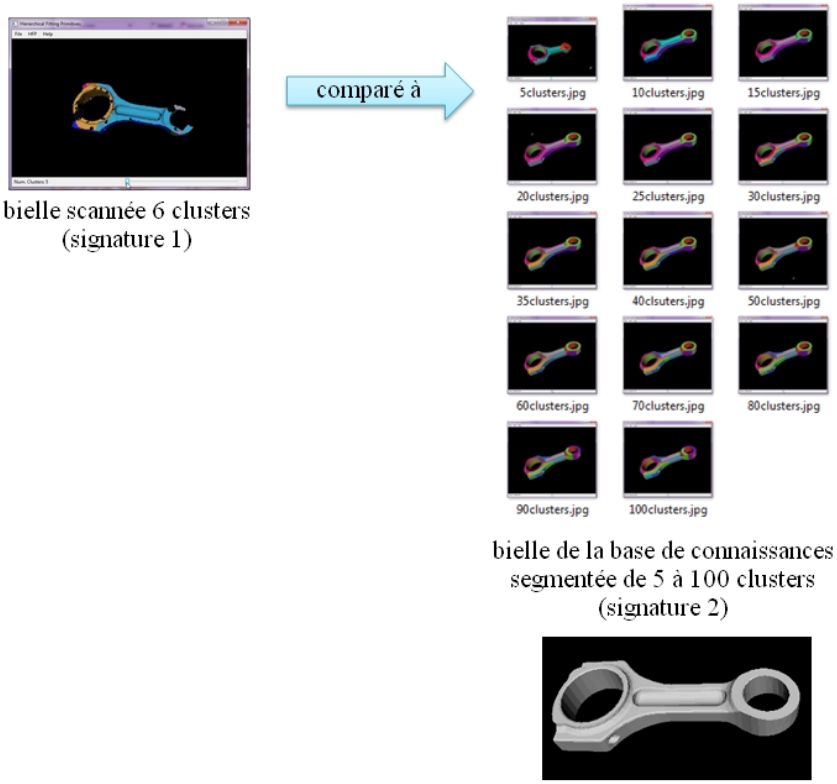


FIGURE B.1 – Ensemble de signatures testées avec la signature de la bielle scannée pour le niveau 1.

6 clusters				
Signature1 (1er jeu de données)	**Signature2 (2eme jeu de données)**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_100clusters.xml;	100	6	11
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_090clusters.xml;	100	6	12
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_080clusters.xml;	100	7	13
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_070clusters.xml;	100	8	15
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_060clusters.xml;	100	10	18
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_050clusters.xml;	100	12	21
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_040clusters.xml;	100	15	26
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_035clusters.xml;	100	17	29
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_030clusters.xml;	100	20	33
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_025clusters.xml;	100	24	38
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_020clusters.xml;	100	30	46
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_015clusters.xml;	100	40	57
signature_niv1_scan_bielle_ifpen_006clusters.xml;	**signature_niv1_bielle_ass_005clusters.xml;**	66	**80**	72
signature_niv1_scan_bielle_ifpen_006clusters.xml;	signature_niv1_bielle_ass_010clusters.xml;	100	60	75

7 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_100clusters.xml	100	7	13
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_090clusters.xml	100	7	14
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_080clusters.xml	100	8	16
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_070clusters.xml	100	10	18
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_060clusters.xml	100	11	20
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_050clusters.xml	100	14	24
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_040clusters.xml	100	17	29
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_035clusters.xml	100	20	33
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_030clusters.xml	100	23	37
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_025clusters.xml	100	28	43
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_020clusters.xml	85	30	44
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_015clusters.xml	85	40	54
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_010clusters.xml	85	60	70
signature_niv1_scan_bielle_ifpen_007clusters.xml	signature_niv1_bielle_ass_005clusters.xml	71	100	83

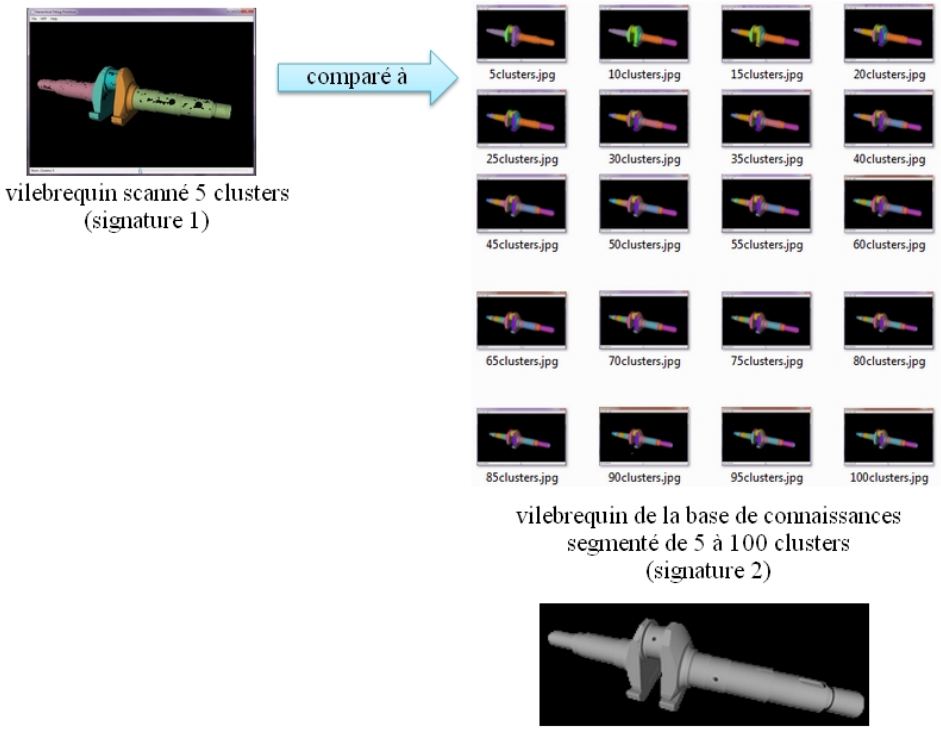
8 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_100clusters.xml	100	8	14
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_090clusters.xml	100	8	16
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_080clusters.xml	100	10	18
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_070clusters.xml	100	11	20
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_060clusters.xml	100	13	23
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_050clusters.xml	100	16	27
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_040clusters.xml	100	20	33
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_035clusters.xml	100	22	37
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_030clusters.xml	100	26	42
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_025clusters.xml	100	32	48
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_020clusters.xml	87	35	50
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_015clusters.xml	87	46	60
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_010clusters.xml	75	60	66
signature_niv1_scan_bielle_ifpen_008clusters.xml	signature_niv1_bielle_ass_005clusters.xml	62	100	76

9 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_100clusters.xml	100	9	16
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_090clusters.xml	100	10	18
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_080clusters.xml	100	11	20
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_070clusters.xml	100	12	22
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_060clusters.xml	100	15	26
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_050clusters.xml	100	18	30
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_040clusters.xml	100	22	36
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_035clusters.xml	100	25	40
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_030clusters.xml	88	26	41
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_025clusters.xml	100	36	52
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_020clusters.xml	88	40	55
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_010clusters.xml	66	60	63
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_015clusters.xml	88	53	66
signature_niv1_scan_bielle_ifpen_009clusters.xml	signature_niv1_bielle_ass_005clusters.xml	55	100	71

10 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_090clusters.xml	90	10	18
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_100clusters.xml	100	10	18
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_080clusters.xml	90	11	20
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_070clusters.xml	90	12	22
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_060clusters.xml	90	15	25
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_050clusters.xml	90	18	30
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_040clusters.xml	90	22	36
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_035clusters.xml	90	25	40
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_030clusters.xml	90	30	45
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_025clusters.xml	90	36	51
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_020clusters.xml	90	45	60
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_005clusters.xml	50	100	66
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_010clusters.xml	70	70	70
signature_niv1_scan_bielle_ifpen_010clusters.xml	signature_niv1_bielle_ass_015clusters.xml	90	60	72

Tableau B.1 – Résultats de la comparaison de niveau 1 pour une bielle scannée et segmentée avec 6, 7, 8, 9 et 10 clusters.

Chaque vilebrequin (signature 1) a été comparé à un jeu de données (signature 2) tel que décrit dans la figure B.2. L'ensemble de tests et résultats sont présentés dans le tableau B.2.



vilebrequin scanné 5 clusters
(signature 1)

vilebrequin de la base de connaissances
segmenté de 5 à 100 clusters
(signature 2)

Figure B.2 – Ensemble de signatures testées avec la signature du vilebrequin pour le niveau 1.

5 clusters				
Signature1 (1er jeu de données)	**Signature2 (2eme jeu de données)**	**Similarité1**	**Similarité2**	**Similarité3**
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_100clusters.xml	100	5	9
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_090clusters.xml	100	5	10
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_095clusters.xml	100	5	10
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_080clusters.xml	100	6	11
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_085clusters.xml	100	5	11
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_075clusters.xml	100	6	12
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_070clusters.xml	100	7	13
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_065clusters.xml	100	7	14
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_060clusters.xml	100	8	15
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_055clusters.xml	100	9	16
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_050clusters.xml	100	10	18
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_045clusters.xml	100	11	20
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_040clusters.xml	100	12	22
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_035clusters.xml	100	14	25
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_030clusters.xml	100	16	28
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_025clusters.xml	100	20	33
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_020clusters.xml	100	25	40
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_015clusters.xml	100	33	50
signature_scan_vilebrequin_005clusters.xml	signature_vilebrequin_bdd_010clusters.xml	100	50	66
signature_scan_vilebrequin_005clusters.xml	**signature_vilebrequin_bdd_005clusters.xml**	80	**80**	80

10 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_100clusters.xml	100	10	18
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_095clusters.xml	100	10	19
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_090clusters.xml	100	11	20
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_085clusters.xml	100	11	21
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_080clusters.xml	100	12	22
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_075clusters.xml	100	13	23
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_070clusters.xml	100	14	25
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_065clusters.xml	100	15	26
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_060clusters.xml	100	16	28
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_055clusters.xml	100	18	30
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_050clusters.xml	100	20	33
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_045clusters.xml	100	22	36
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_040clusters.xml	100	25	40
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_035clusters.xml	100	28	44
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_030clusters.xml	100	33	50
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_025clusters.xml	100	40	57
signature_scan_vilebrequin_010clusters.xml	**signature_vilebrequin_bdd_005clusters.xml**	50	**100**	66
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_020clusters.xml	100	50	66
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_010clusters.xml	80	80	80
signature_scan_vilebrequin_010clusters.xml	signature_vilebrequin_bdd_015clusters.xml	100	66	80

15 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_scan_vilebrequin_015clusters.xml	**signature_vilebrequin_bdd_005clusters**	33	**100**	50
signature_scan_vilebrequin_015clusters.xml	signature_vilebrequin_bdd_010clusters.xml	53	80	64
signature_scan_vilebrequin_015clusters.xml	signature_vilebrequin_bdd_015clusters	73	73	73

20 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_scan_vilebrequin_020clusters.xml	**signature_vilebrequin_bdd_005clusters**	25	**100**	40
signature_scan_vilebrequin_020clusters.xml	signature_vilebrequin_bdd_010clusters	45	90	60
signature_scan_vilebrequin_020clusters.xml	signature_vilebrequin_bdd_015clusters	55	73	62
signature_scan_vilebrequin_020clusters.xml	signature_vilebrequin_bdd_020clusters	55	55	55
signature_scan_vilebrequin_020clusters.xml	signature_vilebrequin_bdd_025clusters	70	56	62

Tableau B.2 – Résultats de la comparaison de niveau 1 pour un vilebrequin scanné et segmenté avec 5, 10, 15 et 20 clusters.

B.2 Résultats des tests de comparaison de niveau 2 pour la bielle et le vilebrequin

Pour le deuxième niveau, le nombre de clusters de la bielle scannée a été ajusté puisqu'il a été possible de comparer des graphes de plus grande taille. Nous retrouvons l'ensemble des résultats dans le tableau B.3.

Pour le vilebrequin, le nombre de clusters de la donnée scannée a été augmenté jusqu'à 65 clusters. Les résultats sont dans les tableaux suivants :

— Tableau B.4 pour le vilebrequin scanné de 5 à 20 clusters ;
— Tableau B.5 pour le vilebrequin scanné de 25 à 40 clusters ;
— Tableau B.6 pour le vilebrequin scanné de 45 à 60 clusters ;
— Tableau B.7 pour le vilebrequin scanné de 65 clusters.

5 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_100clusters.xml	0	0	0
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_80clusters.xml	0	0	0
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_90clusters.xml	40	2	4
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_70clusters.xml	60	4	8
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_60clusters.xml	60	5	9
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_50clusters.xml	80	8	14
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_40clusters.xml	80	10	17
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_35clusters.xml	80	11	20
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_30clusters.xml	80	13	22
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_25clusters.xml	80	16	26
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_20clusters.xml	100	25	40
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_15clusters.xml	100	33	50
signature_niv2_scan_bielle_ifpen_5clusters.xml	signature_niv2_bielle_ass_10clusters.xml	80	40	53
signature_niv2_scan_bielle_ifpen_5clusters.xml	**signature_niv2_bielle_ass_5clusters.xml**	80	**80**	80

10 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_100clusters.xml	0	0	0
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_80clusters.xml	0	0	0
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_90clusters.xml	20	2	4
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_60clusters.xml	30	5	8
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_70clusters.xml	40	5	10
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_40clusters.xml	40	10	16
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_50clusters.xml	60	12	20
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_35clusters.xml	70	20	31
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_30clusters.xml	80	26	40
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_10clusters.xml	50	50	50
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_25clusters.xml	90	36	51
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_20clusters.xml	80	40	53
signature_niv2_scan_bielle_ifpen_10clusters.xml	**signature_niv2_bielle_ass_5clusters.xml**	40	**80**	53
signature_niv2_scan_bielle_ifpen_10clusters.xml	signature_niv2_bielle_ass_15clusters.xml	70	46	56

15 clusters				
Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_100clusters.xml	20	3	5
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_80clusters.xml	26	5	8
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_90clusters.xml	33	5	9
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_70clusters.xml	40	8	14
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_60clusters.xml	46	11	18
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_50clusters.xml	60	18	27
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_40clusters.xml	53	20	29
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_35clusters.xml	66	28	40
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_30clusters.xml	73	36	48
signature_niv2_scan_bielle_ifpen_15clusters.xml	**signature_niv2_bielle_ass_5clusters.xml**	33	**100**	50
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_25clusters.xml	73	44	55
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_10clusters.xml	46	70	56
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_15clusters.xml	60	60	60
signature_niv2_scan_bielle_ifpen_15clusters.xml	signature_niv2_bielle_ass_20clusters.xml	73	55	62

20 clusters				
Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_100clusters.xml	25	5	8
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_90clusters.xml	35	7	12
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_80clusters.xml	35	8	14
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_70clusters.xml	45	12	20
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_60clusters.xml	50	16	25
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_50clusters.xml	50	20	28
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_40clusters.xml	55	27	36
signature_niv2_scan_bielle_ifpen_20clusters.xml	**signature_niv2_bielle_ass_5clusters.xml**	25	**100**	40
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_35clusters.xml	60	34	43
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_10clusters.xml	35	70	46
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_15clusters.xml	45	60	51
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_30clusters.xml	65	43	52
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_25clusters.xml	60	48	53
signature_niv2_scan_bielle_ifpen_20clusters.xml	signature_niv2_bielle_ass_20clusters.xml	55	55	55

Tableau B.3 – Résultats de la comparaison de niveau 2 pour une bielle scannée segmentée avec 5, 10, 15 et 20 clusters.

5 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	0	0	0
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	40	3	6
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	40	4	8
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	60	4	8
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	60	5	10
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	60	8	8
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	60	10	17
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	80	10	17
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	100	10	18
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	100	20	33
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	100	25	40
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	100	33	50
signature_niv2_scan_vilebrequin_5clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	100	50	66
signature_niv2_scan_vilebrequin_5clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	80	**80**	80

10 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	20	2	3
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	20	2	3
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	20	2	4
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	20	2	4
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	20	2	4
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	20	2	4
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	20	3	5
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	30	5	8
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	30	6	10
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	40	5	10
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	40	7	12
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	40	11	17
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	40	13	20
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	50	12	20
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	70	14	23
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	60	30	40
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	70	28	40
signature_niv2_scan_vilebrequin_10clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	50	**100**	66
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	90	60	72
signature_niv2_scan_vilebrequin_10clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	80	80	80

15 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	20	3	5
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	20	3	5
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	20	3	5
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	20	4	6
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	20	3	6
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	20	3	6
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	26	6	10
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	26	6	10
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	33	7	11
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	33	9	14
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	33	11	16
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	33	12	18
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	46	14	21
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	46	20	28
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	46	23	31
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	53	32	40
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	53	40	45
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	40	60	48
signature_niv2_scan_vilebrequin_15clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	33	**100**	50
signature_niv2_scan_vilebrequin_15clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	60	60	60

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	15	3	5
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	15	3	5
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	15	3	5
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	15	3	5
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	15	3	6
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	20	5	8
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	20	6	9
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	25	8	12
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	25	9	13
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	30	8	13
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	25	12	16
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	30	13	18
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	35	14	20
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	35	20	25
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	35	23	28
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	40	32	35
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	40	40	40
signature_niv2_scan_vilebrequin_20clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	25	**100**	40
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	45	60	51
signature_niv2_scan_vilebrequin_20clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	40	80	53

Tableau B.4 – Résultats de la comparaison de niveau 2 pour un vilebrequin scanné et segmenté avec 5, 10, 15 et 20 clusters.

25clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	12	3	5
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	12	3	5
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	16	4	6
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	16	5	7
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	16	4	7
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	16	6	8
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	16	5	8
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	20	7	10
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	20	8	11
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	20	12	15
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	28	12	17
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	28	15	20
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	36	18	24
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	28	23	25
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	36	25	30
signature_niv2_scan_vilebrequin_25clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	20	**100**	33
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	40	40	40
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	40	50	44
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	32	80	45
signature_niv2_scan_vilebrequin_25clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	40	66	50

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	13	4	6
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	16	5	7
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	16	5	8
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	16	5	8
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	20	7	10
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	20	8	12
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	23	9	13
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	23	10	14
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	23	11	15
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	20	13	16
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	20	15	17
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	20	20	20
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	23	20	21
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	30	16	21
signature_niv2_scan_vilebrequin_30clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	16	**100**	28
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	40	24	30
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	26	40	32
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	30	36	32
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	26	53	35
signature_niv2_scan_vilebrequin_30clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	26	80	40

35 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	17	6	8
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	14	5	8
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	17	6	9
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	17	6	9
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	17	7	10
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	17	8	11
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	20	11	14
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	22	10	14
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	20	17	18
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	25	13	18
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	22	17	20
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	25	16	20
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	20	23	21
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	28	20	23
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	25	25	25
signature_niv2_scan_vilebrequin_35clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	14	**100**	25
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	20	46	28
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	25	36	30
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	20	70	31
signature_niv2_scan_vilebrequin_35clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	28	50	36

40 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	15	6	8
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	15	7	9
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	15	6	9
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	17	7	10
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	15	7	10
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	17	10	12
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	20	10	13
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	15	15	15
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	20	12	15
signature_niv2_scan_vilebrequin_40clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	10	**80**	17
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	22	16	18
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	22	15	18
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	17	23	20
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	15	40	21
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	25	22	23
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	15	60	24
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	27	22	24
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	25	28	26
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	27	44	33
signature_niv2_scan_vilebrequin_40clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	27	55	36

Tableau B.5 – Résultats de la comparaison de niveau 2 pour un vilebrequin scanné et segmenté avec 25, 30, 35 et 40 clusters.

45 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	20	9	12
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	20	10	13
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	20	10	13
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	17	12	14
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	24	11	15
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	17	14	16
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	22	13	16
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	22	12	16
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	17	17	17
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	20	15	17
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	17	20	18
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	20	18	18
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	24	15	19
signature_niv2_scan_vilebrequin_45clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	11	**100**	20
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	13	60	21
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	17	26	21
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	20	25	22
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	15	46	23
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	20	45	27
signature_niv2_scan_vilebrequin_45clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	24	44	31

50 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	16	8	10
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	16	10	12
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	18	9	12
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	18	10	13
signature_niv2_scan_vilebrequin_50clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	8	**80**	14
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	18	12	14
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	20	11	14
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	14	17	15
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	18	13	15
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	18	12	15
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	10	50	16
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	20	16	18
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	20	20	20
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	20	22	21
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	18	36	24
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	26	23	24
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	18	45	25
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	18	60	27
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	24	40	30
signature_niv2_scan_vilebrequin_50clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	28	40	32

55 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	16	9	11
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	14	10	11
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	16	10	12
signature_niv2_scan_vilebrequin_55clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	7	**80**	13
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	16	12	13
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	18	11	13
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	18	10	13
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	16	12	14
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	9	50	15
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	16	13	15
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	14	17	16
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	16	22	18
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	12	46	20
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	20	20	20
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	21	20	20
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	14	40	21
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	16	36	22
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	23	26	24
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	20	36	25
signature_niv2_scan_vilebrequin_55clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	25	40	31

60 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	15	9	11
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	6	40	11
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	13	10	11
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	13	9	11
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	15	9	11
signature_niv2_scan_vilebrequin_60clusters.xml	**signature_niv2_vilebrequin_bdd_5clusters.xml**	6	**80**	12
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	15	10	12
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	11	17	14
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	13	17	15
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	16	15	16
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	18	15	16
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	18	14	16
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	18	20	19
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	18	22	20
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	15	36	21
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_20clusters.xml	15	45	22
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	16	33	22
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	23	23	23
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	15	60	24
signature_niv2_scan_vilebrequin_60clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	21	37	27

Tableau B.6 – Résultats de la comparaison de niveau 2 pour un vilebrequin scanné et segmenté avec 45, 50, 55 et 60 clusters.

65 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_10clusters.xml	3	20	5
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_5clusters.xml	3	40	5
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_15clusters.xml	7	33	12
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_85clusters.xml	13	10	12
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_90clusters.xml	15	11	12
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_100clusters.xml	16	11	13
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_80clusters.xml	15	12	13
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_95clusters.xml	16	11	13
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_70clusters.xml	15	14	14
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_55clusters.xml	15	18	16
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_65clusters.xml	16	16	16
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_40clusters.xml	13	22	17
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_50clusters.xml	15	20	17
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_45clusters.xml	15	22	18
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_75clusters.xml	20	17	18
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_25clusters.xml	13	36	20
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_60clusters.xml	20	21	20
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_30clusters.xml	15	33	21
signature_niv2_scan_vilebrequin_65clusters.xml	signature_niv2_vilebrequin_bdd_35clusters.xml	16	31	22
signature_niv2_scan_vilebrequin_65clusters.xml	**signature_niv2_vilebrequin_bdd_20clusters.xml**	15	**50**	23

Tableau B.7 – Résultats de la comparaison de niveau 2 pour un vilebrequin scanné et segmenté avec 65 clusters.

B.3 Résultats des tests de comparaison pour les graphes de niveau 1 sur un ensemble de données

Cette série de tests consiste à comparer la donnée scannée à un ensemble de données de la base de connaissances. La figure B.3 illustre le principe des tests avec le deuxième jeu de données dont le nombre de clusters est fixé, c'est-à-dire variable 2 = 15 clusters.

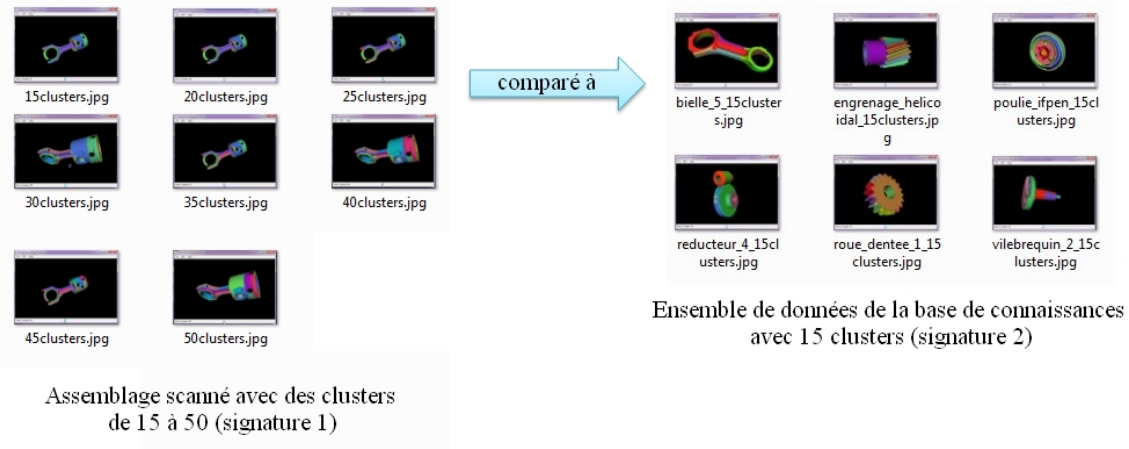


Figure B.3 – Ensemble de données utilisées pour la première série de tests avec 15 clusters

L'ensemble des résultats pour "variable 2 = 15 clusters" est présenté dans le tableau B.8.

Pour la deuxième série de tests (variable 2 = 20 clusters), les résultats sont dans le tableau B.9.

Pour la troisième série de tests ((variable 2 = 25 clusters)), les résultats sont dans le tableau B.10.

0 à 5 min | 6 à 10 min | 11 à 30 min | 31 min à 5h | > 5h

15 clusters

Signature1 (1er jeu de données)	Signature2 (2eme jeu de données)	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_scan_bielle_piston_15clusters.xml	signature_niv1_bielle_5_15clusters.xml	53	53	53	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_engrenage_helicoidal_15clusters.xml	60	60	60	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_roue_dentee_1_15clusters.xml	60	60	60	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_poulie_ifpen_15clusters.xml	66	66	66	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_vilebrequin_2_15clusters.xml	66	66	66	< 1 min
signature_scan_bielle_piston_15clusters.xml	signature_niv1_reducteur_4_15clusters.xml	73	73	73	< 1 min

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_scan_bielle_piston_20clusters.xml	signature_niv1_bielle_5_15clusters.xml	45	60	51	< 10 min
signature_scan_bielle_piston_20clusters.xml	signature_niv1_engrenage_helicoidal_15clusters.xml	55	73	62	< 5 min
signature_scan_bielle_piston_20clusters.xml	signature_niv1_roue_dentee_1_15clusters.xml	55	73	62	< 30 min
signature_scan_bielle_piston_20clusters.xml	signature_niv1_vilebrequin_2_15clusters.xml	55	73	62	< 10 min
signature_scan_bielle_piston_20clusters.xml	signature_niv1_poulie_ifpen_15clusters.xml	60	80	68	< 10 min
signature_scan_bielle_piston_20clusters.xml	signature_niv1_reducteur_4_15clusters.xml	60	80	68	< 10 min

25 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_bielle_5_15clusters.xml	40	66	50	< 15 min
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_engrenage_helicoidal_15clusters.xml	44	73	55	< 22 min
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_roue_dentee_1_15clusters.xml	44	73	55	< 4h40
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_vilebrequin_2_15clusters.xml	48	80	60	< 2h40
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_poulie_ifpen_15clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_reducteur_4_15clusters.xml				> 5h

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_engrenage_helicoidal_15clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_roue_dentee_1_15clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_vilebrequin_2_15clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_reducteur_4_15clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_bielle_5_15clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_poulie_ifpen_15clusters.xml				> 5h

Tableau B.8 – Résultats de la comparaison de niveau 1 pour un ensemble de données diverses (variable 2 = 15 clusters).

15 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_scan_bielle_piston_15clusters.xml	signature_engrenage_helicoidal_20clusters.xml	60	45	51	< 30 min
signature_scan_bielle_piston_15clusters.xml	signature_roue_dentee_1_20clusters.xml	60	45	51	< 2 min
signature_scan_bielle_piston_15clusters.xml	signature_bielle_5_light_20clusters.xml	73	55	62	< 2 min
signature_scan_bielle_piston_15clusters.xml	signature_poulie_ifpen_20clusters.xml	73	55	62	< 15 min
signature_scan_bielle_piston_15clusters.xml	signature_reducteur_4_20clusters.xml	73	55	62	< 2 min
signature_scan_bielle_piston_15clusters.xml	signature_vilebrequin_2_20clusters.xml	80	60	68	< 20 min

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_bielle_5_light_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_engrenage_helicoidal_20clusters.xml	60	60	60	< 45 min
signature_niv1_scan_bielle_piston_20clusters.xml	signature_poulie_ifpen_20clusters.xml	65	65	65	< 2h
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_reducteur_4_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_roue_dentee_1_20clusters.xml	60	60	60	< 40 min
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_vilebrequin_2_20clusters.xml				> 5h

25 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_engrenage_helicoidal_20clusters.xml	56	70	62	< 2h30
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_roue_dentee_1_20clusters.xml	56	70	62	< 2h15
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_bielle_5_light_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_poulie_ifpen_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_reducteur_4_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_vilebrequin_2_20clusters.xml				> 5h

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_engrenage_helicoidal_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_roue_dentee_1_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_bielle_5_light_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_poulie_ifpen_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_reducteur_4_20clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_vilebrequin_2_20clusters.xml				> 5h

Tableau B.9 – Résultats de la comparaison de niveau 1 pour un ensemble de données diverses (variable 2 = 20 clusters).

15 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_engrenage_helicoidal_25clusters.xml	60	36	45	< 6 min
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_roue_dentee_1_25clusters.xml	60	36	45	< 4 min
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_poulie_ifpen_25clusters.xml	73	44	55	< 5 min
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_reducteur_4_25clusters.xml	80	48	60	< 1h22
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_bielle_5_light_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_15clusters.xml	signature_niv1_vilebrequin_2_25clusters.xml				> 5h

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_engrenage_helicoidal_25clusters.xml	60	48	53	< 2h13
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_roue_dentee_1_25clusters.xml	60	48	53	< 1h57
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_poulie_ifpen_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_reducteur_4_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_vilebrequin_2_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_20clusters.xml	signature_niv1_bielle_5_light_25clusters.xml				> 5h

25 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_engrenage_helicoidal_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_roue_dentee_1_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_poulie_ifpen_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_reducteur_4_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_vilebrequin_2_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_25clusters.xml	signature_niv1_bielle_5_light_25clusters.xml				> 5h

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_roue_dentee_1_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_engrenage_helicoidal_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_reducteur_4_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_vilebrequin_2_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_poulie_ifpen_25clusters.xml				> 5h
signature_niv1_scan_bielle_piston_30clusters.xml	signature_niv1_bielle_5_light_25clusters.xml				> 5h

Tableau B.10 – Résultats de la comparaison de niveau 1 pour un ensemble de données diverses (variable 2 = 25 clusters).

B.4 Résultats des tests de comparaison pour les graphes de niveau 2 sur un ensemble de données

Pour ce deuxième niveau, les jeux de données restent identiques. Seules les signatures de niveau 2 sont utilisées.

Les résultats complets sont dans les tableaux suivants :

— pour variable 2 = 15 clusters : Tableau B.11 et Tableau B.12
— pour variable 2 = 20 clusters : Tableau B.13 et Tableau B.14
— pour variable 2 = 25 clusters : Tableau B.15 et Tableau B.16

15 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	33	33	33	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_reducteur_4_15clusters.xml	46	46	46	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_bielle_5_15clusters.xml	53	53	53	< 10s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	60	60	60	< 10s

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	30	40	34	< 10s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_reducteur_4_15clusters.xml	40	53	45	< 10s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_bielle_5_15clusters.xml	45	60	51	< 10s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	50	66	57	< 10s

25 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 30 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 30 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	24	40	30	< 2 min
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_reducteur_4_15clusters.xml	32	53	40	< 20s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_bielle_5_15clusters.xml	40	66	50	< 20s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	48	80	60	< 20s

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	20	40	26	< 10s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_reducteur_4_15clusters.xml	20	40	26	< 10s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_bielle_5_15clusters.xml	26	53	35	< 10s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	26	53	35	< 10s

Tableau B.11 – Résultats de la comparaison de niveau 2 (variable 2 = 15 clusters et variable 1 = 15,20,25,30).

35 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 1 min
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 20s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	17	40	24	< 20s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_reducteur_4_15clusters.xml	22	53	32	< 20s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_bielle_5_15clusters.xml	28	66	40	< 2h
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	37	86	52	< 55 min

40 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 10s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	12	33	18	< 10s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_reducteur_4_15clusters.xml	12	33	18	< 10s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_bielle_5_15clusters.xml	17	46	25	< 10s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	20	53	29	< 10s

45 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	8	26	13	< 1 min
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	8	26	13	< 20 s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	13	40	20	< 20 s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_reducteur_4_15clusters.xml	17	53	26	< 20 s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	28	86	43	< 48 min
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_bielle_5_15clusters.xml				> 5h

50 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_engrenage_helicoidal_15clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_roue_dentee_1_15clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_poulie_ifpen_15clusters.xml	10	33	15	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_reducteur_4_15clusters.xml	12	40	18	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_bielle_5_15clusters.xml	14	46	21	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_vilebrequin_2_15clusters.xml	14	46	21	< 10 s

Tableau B.12 – Résultats de la comparaison de niveau 2 (variable 2 = 15 clusters et variable 1 = 35,40,45,50).

15 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_reducteur_4_20clusters.xml	26	20	22	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	33	25	28	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	53	40	45	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	73	55	62	< 10 s

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	0	0	0	< 3h
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 30 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_reducteur_4_20clusters.xml	25	25	25	< 30 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	25	25	25	< 15s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	50	50	50	< 15s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	60	60	60	< 15s

25 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	0	0	0	< 16 min
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 15 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_reducteur_4_20clusters.xml	20	25	22	< 15 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	24	30	26	< 15 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	52	65	57	< 15 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_bielle_5_light_20clusters.xml				> 5h

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	10	15	12	< 10 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	20	30	24	< 10 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_reducteur_4_20clusters.xml	20	30	24	< 10 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	20	30	24	< 10 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	33	50	40	< 10 s

Tableau B.13 – Résultats de la comparaison de niveau 2 (variable 2 = 20 clusters et variable 1 = 15,20,25,30).

35 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	0	0	0	< 1 min
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 15 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_reducteur_4_20clusters.xml	14	25	18	< 15 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	17	30	21	< 15 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	37	65	47	< 2h30
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_bielle_5_light_20clusters.xml				> 5h

40 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	5	10	6	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	15	30	20	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_reducteur_4_20clusters.xml	15	30	20	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	15	30	20	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	22	45	30	< 10 s

45 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	8	20	12	< 1 min
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	8	20	12	< 20 s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_reducteur_4_20clusters.xml	17	40	24	< 20 s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	20	45	27	< 20 s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	28	65	40	< 4h17
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_bielle_5_light_20clusters.xml				> 5h

50 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_roue_dentee_1_20clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_engrenage_helicoidal_20clusters.xml	4	10	5	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_reducteur_4_20clusters.xml	10	25	14	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_poulie_ifpen_20clusters.xml	12	30	17	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_vilebrequin_2_20clusters.xml	14	35	20	< 10 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_bielle_5_light_20clusters.xml	16	40	22	< 10 s

Tableau B.14 – Résultats de la comparaison de niveau 2 (variable 2 = 20 clusters et variable 1 = 35,40,45,50).

15 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	26	16	20	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_reducteur_4_25clusters.xml	40	24	30	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	60	36	45	< 10 s
signature_niv2_scan_bielle_piston_15clusters.xml	signature_niv2_bielle_5_light_25clusters.xml	80	48	60	< 10 s

20 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	0	0	0	< 20 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 20 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	25	20	22	< 20 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_reducteur_4_25clusters.xml	40	32	35	< 20 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	55	44	48	< 20 s
signature_niv2_scan_bielle_piston_20clusters.xml	signature_niv2_bielle_5_light_25clusters.xml				> 5h

25 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	0	0	0	< 1 min
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 20 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	24	24	24	< 20 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_reducteur_4_25clusters.xml	36	36	36	< 20 s
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	56	56	56	< 43 min
signature_niv2_scan_bielle_piston_25clusters.xml	signature_niv2_bielle_5_light_25clusters.xml				> 5h

30 clusters

Signature1	Signature2	Similarité1	Similarité2	Similarité3	Temps de calcul
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 20 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	6	8	7	< 20 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_reducteur_4_25clusters.xml	13	16	14	< 20 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	16	20	18	< 20 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	23	28	25	< 20 s
signature_niv2_scan_bielle_piston_30clusters.xml	signature_niv2_bielle_5_light_25clusters.xml				> 5h

Tableau B.15 – Résultats de la comparaison de niveau 2 (variable 2 = 25 clusters et variable 1 = 15,20,25,30).

35 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	0	0	0	< 20 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 20 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	14	20	16	< 20 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_reducteur_4_25clusters.xml	22	32	26	< 20 s
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml				> 5h
signature_niv2_scan_bielle_piston_35clusters.xml	signature_niv2_bielle_5_light_25clusters.xml				> 5h

40 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	5	8	6	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_reducteur_4_25clusters.xml	10	16	12	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	15	24	18	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	17	28	21	< 10 s
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_bielle_5_light_25clusters.xml	25	40	30	< 10 s

45 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	8	16	11	< 1 min
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	8	16	11	< 30s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	13	24	17	< 30s
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_reducteur_4_25clusters.xml	17	32	22	< 2 min
signature_niv2_scan_bielle_piston_45clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml				> 5h
signature_niv2_scan_bielle_piston_40clusters.xml	signature_niv2_bielle_5_light_25clusters.xml				> 5h

50 clusters				
Signature1	**Signature2**	**Similarité1**	**Similarité2**	**Similarité3**	**Temps de calcul**
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_roue_dentee_1_25clusters.xml	0	0	0	< 12 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_engrenage_helicoidal_25clusters.xml	4	8	5	< 12 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_reducteur_4_25clusters.xml	8	16	10	< 12 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_vilebrequin_2_25clusters.xml	12	24	16	< 1 min
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_poulie_ifpen_25clusters.xml	14	28	18	< 12 s
signature_niv2_scan_bielle_piston_50clusters.xml	signature_niv2_bielle_5_light_25clusters.xml	18	36	24	< 12 s

Tableau B.16 – Résultats de la comparaison de niveau 2 (variable 2 = 25 clusters et variable 1 = 35,40,45,50).

ANNEXE C

Interfaces logicielles

C.1 Interfaces MATLAB pour générer la signature de niveau 3 en base

Le but de cette section est de présenter les interfaces qui ont été développées afin de générer le graphe $GPfus$ et ceci grâce à la fusion (ou mise en correspondance) de G'_2 et G'_3.

Première interface Elle aide l'utilisateur à mettre en correspondance le graphe de précédence et les surfaces correspondantes dans le maillage. Nous allons prendre l'exemple des plans P1 et P2 liés dans le graphe G'_3 de la figure 3.26. Ils sont liés par une contrainte de parallélisme. Ce plan est référencé à l'aide d'une mise en plan de la pièce qui est présentée dans la figure C.1.

Afin d'identifier à quels numéros de nœud (numéro de région dans le fichier XML) correspondent P1 et P2, la première interface permet d'afficher le maillage relatif à la pièce. Ce maillage a été généré à partir du modèle CAO de la pièce en le convertissant en STL. Puis le maillage est segmenté en un nombre de clusters au moins égal au nombre de faces (de la CAO) et ses signatures de niveaux 1 puis 2 sont générées. Pour cette étape, nous utiliserons la signature de niveau 2 et chaque cluster (région) s'affichera dans l'interface. La figure C.2 illustre cette dernière.

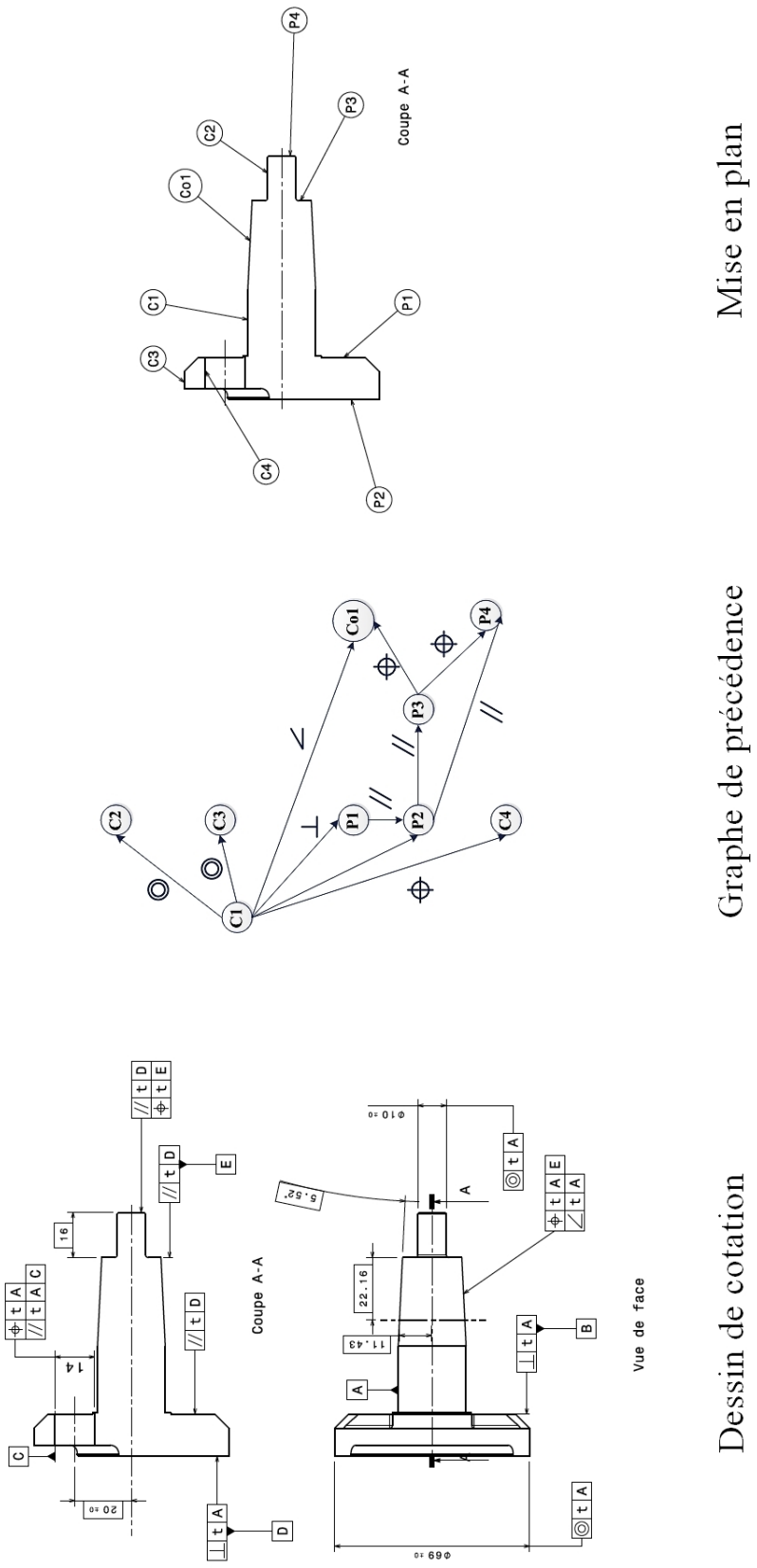


Figure C.1 – Ensemble des données nécessaires pour mettre en correspondance G'_2 et G'_3. : à gauche le dessin de cotation GPS, au milieu le graphe de précédence (G'_3) et à droite une mise en plan avec les identifiants relatifs au graphe de précédence.

Le bouton n°1 permet de lancer l'algorithme afin d'afficher le nuage de points (maillage). Cette opération est réalisée grâce à la lecture conjointe d'un des fichiers XML de signature ainsi que du fichier XML issu de la scène Efpisoft (ce fichier contient les coordonnées de tous les points ainsi que leur appartenance à chaque région/segment).

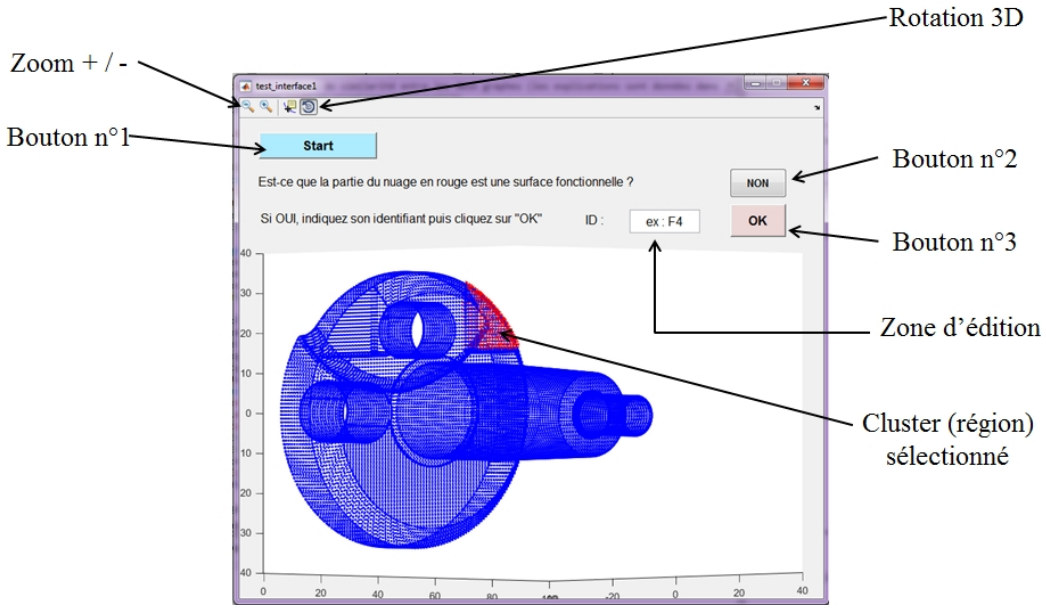


FIGURE C.2 – Présentation de la première interface permettant l'affichage du maillage (développée sous MATLAB R2014b).

L'utilisateur vérifie visuellement si la région s'affichant en rouge est une surface fonctionnelle en s'aidant des mises en plan (figure C.1). Si c'est le cas, il renseigne la zone d'édition en ajoutant l'identifiant de la région (ex : C1). Puis il valide son entrée grâce au bouton n°3 ("OK"). Dans le cas où la région en rouge n'est pas une surface fonctionnelle (c'est-à-dire qu'elle n'est pas référencée dans la figure C.1), l'utilisateur appuie sur le bouton n°2 ("NON"). Dans l'exemple de la figure C.2, la région en rouge ne correspond à aucune surface fonctionnelle référencée.

Chaque région va ainsi défiler jusqu'à ce qu'on arrive à identifier P1 et P2. La figure C.3 illustre cela. Une fois que toutes les régions ont défilé et que toutes les surfaces fonctionnelles ont été identifiées par l'utilisateur, cette interface se ferme et une deuxième s'ouvre automatiquement.

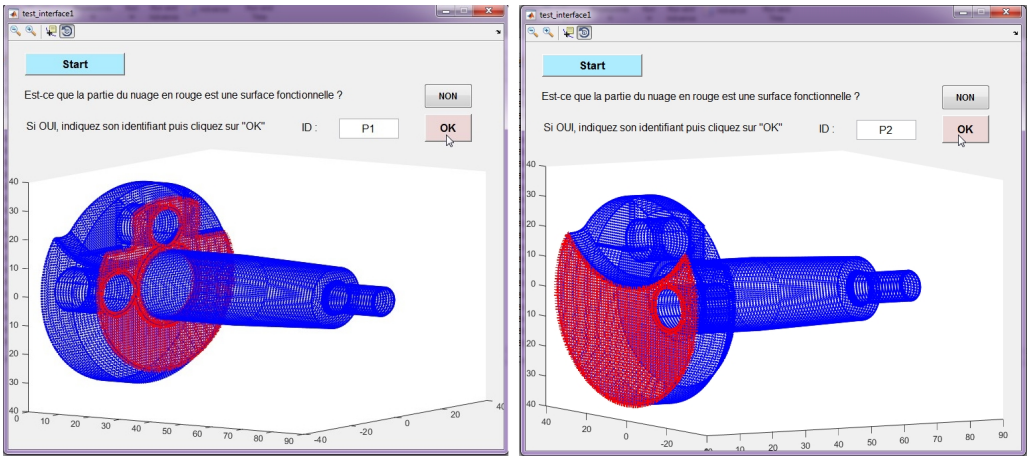


FIGURE C.3 – Affichage des plans P1 et P2 correspondant au graphe de précédence du vilebrequin.

Deuxième interface Elle permet de renseigner les contraintes fonctionnelles entre les surfaces (régions).

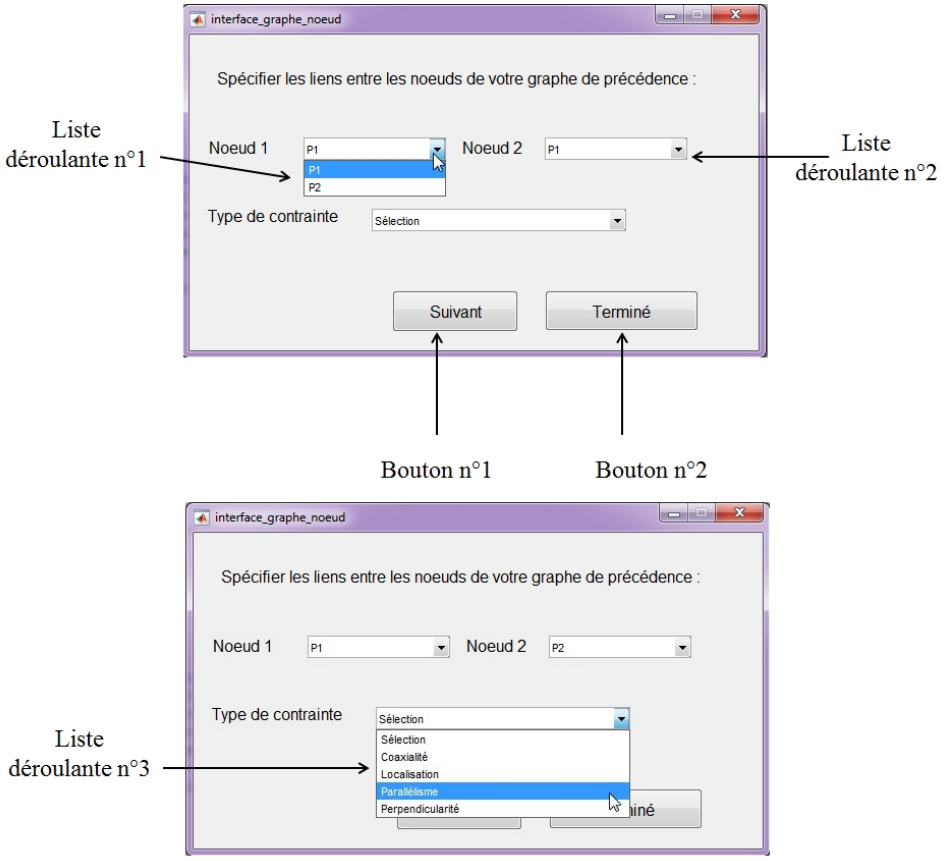


Figure C.4 – Présentation de la deuxième interface afin de renseigner les contraintes fonctionnelles (développée sous MATLAB R2014b).

L'utilisateur sélectionne l'identifiant du premier nœud dans la liste déroulante n°1 puis le deuxième identifiant dans la liste déroulante 2. Puis il sélectionne la contrainte correspondante dans la liste déroulante n°3. Cette opération est réalisée par l'utilisateur et par lecture de chaque contrainte du graphe de précédence (d'après la figure C.1). Notons que les listes déroulantes n°1 et 2 contiennent uniquement les identifiants des nœuds qui ont été renseignés dans la première interface. Dans notre exemple, nous avons uniquement entré P1 et P2.

Après chaque contrainte ajoutée, l'utilisateur valide son choix à l'aide du bouton n°1 ("Suivant"). Puis lorsqu'il arrive à la dernière contrainte, il clique sur le bouton n°2 ("Terminé"). Dans notre exemple, nous cliquerons directement sur "Terminé" car nous renseignerons qu'une seule contrainte (un parallélisme).

Suite à cela, le XML de signature de niveau 3 est généré automatiquement. Il contient toutes les informations initiales de la signature de niveau 2 ainsi qu'une partie encadrée des balises "PrecedenceGraph" qui lui est ajoutée à la fin.

Un aperçu du fichier XML relatif à la signature de niveau 3 est présenté dans la figure C.5. Nous y retrouvons les plans P1 et P2 ainsi que la contrainte fonctionnelle qui les lie.

```
<PrecedenceGraph version="1.0">
   <link functionalSurfaceNum1="1" functionalSurfaceNum2="2" geometricCharacteristic="Parallelisme"/>
</PrecedenceGraph>
```

Figure C.5 – Affichage du code XML de la signature de niveau 3 du vilebrequin.

C.2 Interfaces METIS

Cette section a pour but de présenter brièvement les interfaces du démonstrateur METIS qui est en cours de développement au sein de la société DeltaCAD. Les principales fonctions seront exposées dans les sous-sections suivantes.

Import des données, visualisation et édition

Il est possible d'importer des données de différents types, dites données hétérogènes : 3d, texte, image etc. Puis la visualisation et l'édition de certaines d'entres elles peut être réalisé. La figure C.6 illustre ces fonctions.

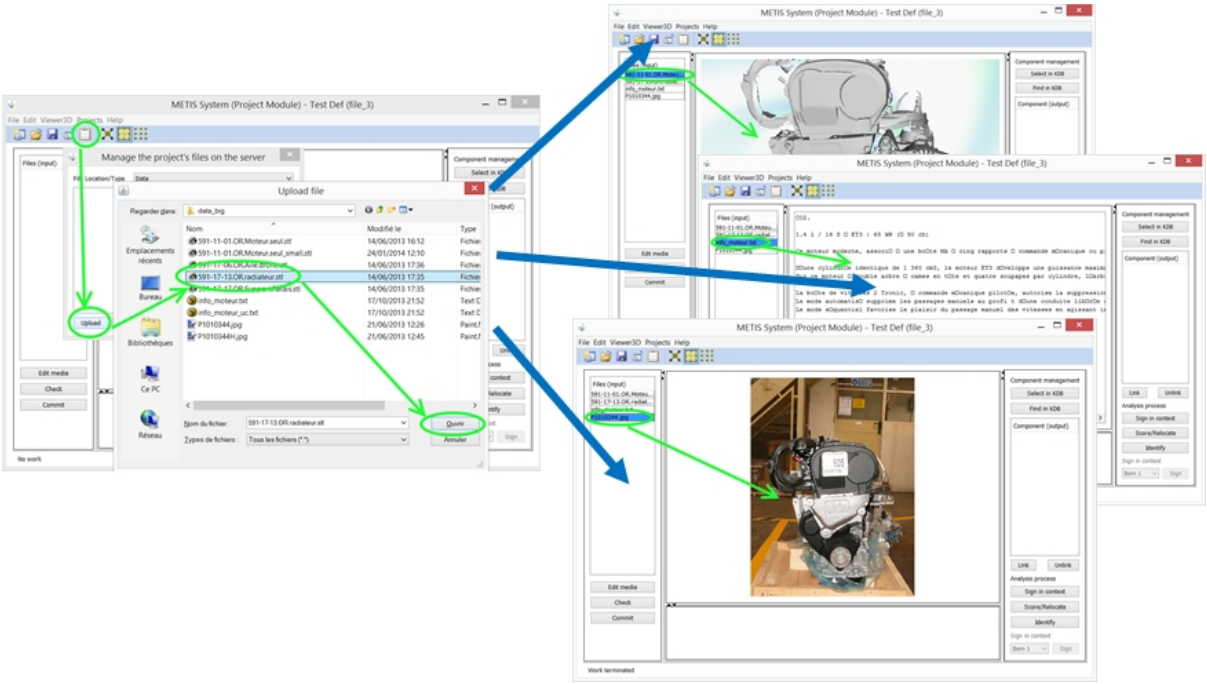


Figure C.6 – Illustration des fonctions d'importation, de visualisation et d'édition dans l'interface METIS (développement de l'interface réalisé par DeltaCAD).

Il est également possible d'importer des données de la base de connaissances. Ces données peuvent être des composants ayant déjà été identifiés dans des processus de RE antérieurs comme des *templates* CAO. Li figure C.7 illustre cela.

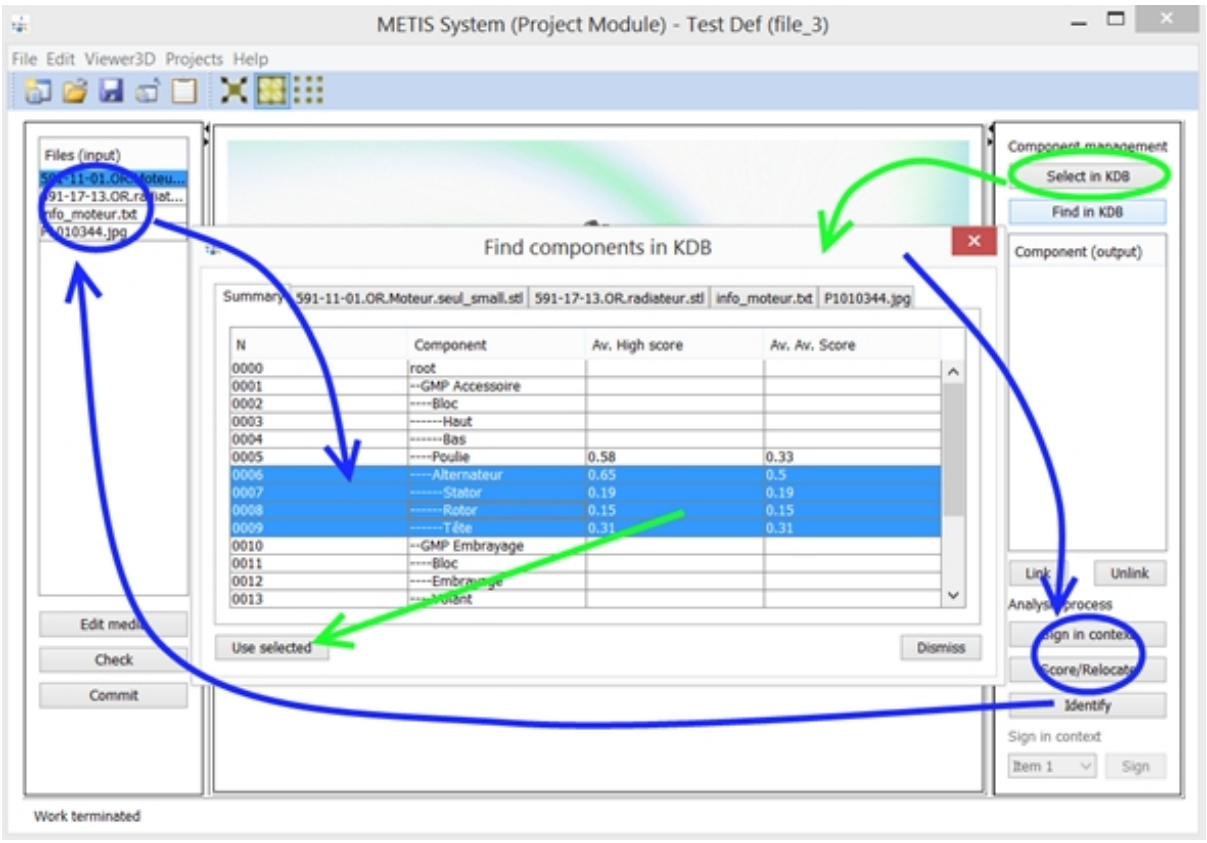


Figure C.7 – Importation de données depuis la base de connaissances dans l'interface METIS (développement de l'interface réalisé par DeltaCAD).

Signature et *scoring*

Lors de l'importation des données, ces dernières sont localisées manuellement en spécifiant le contexte de RE. Cela peut consister, par exemple, à créer la nomenclature de l'assemblage puis à y positionner la donnée d'entrée. Puis les données sont signées. L'étape de *scoring* est ce que nous avons appelé "comparaison des signatures" dans notre manuscrit. Ces scores sont des valeurs de similitude. Une fois ces valeurs affichées, la donnée signée est, si nécessaire, relocalisée dans la nomenclature. La figure C.8 présente ces opérations.

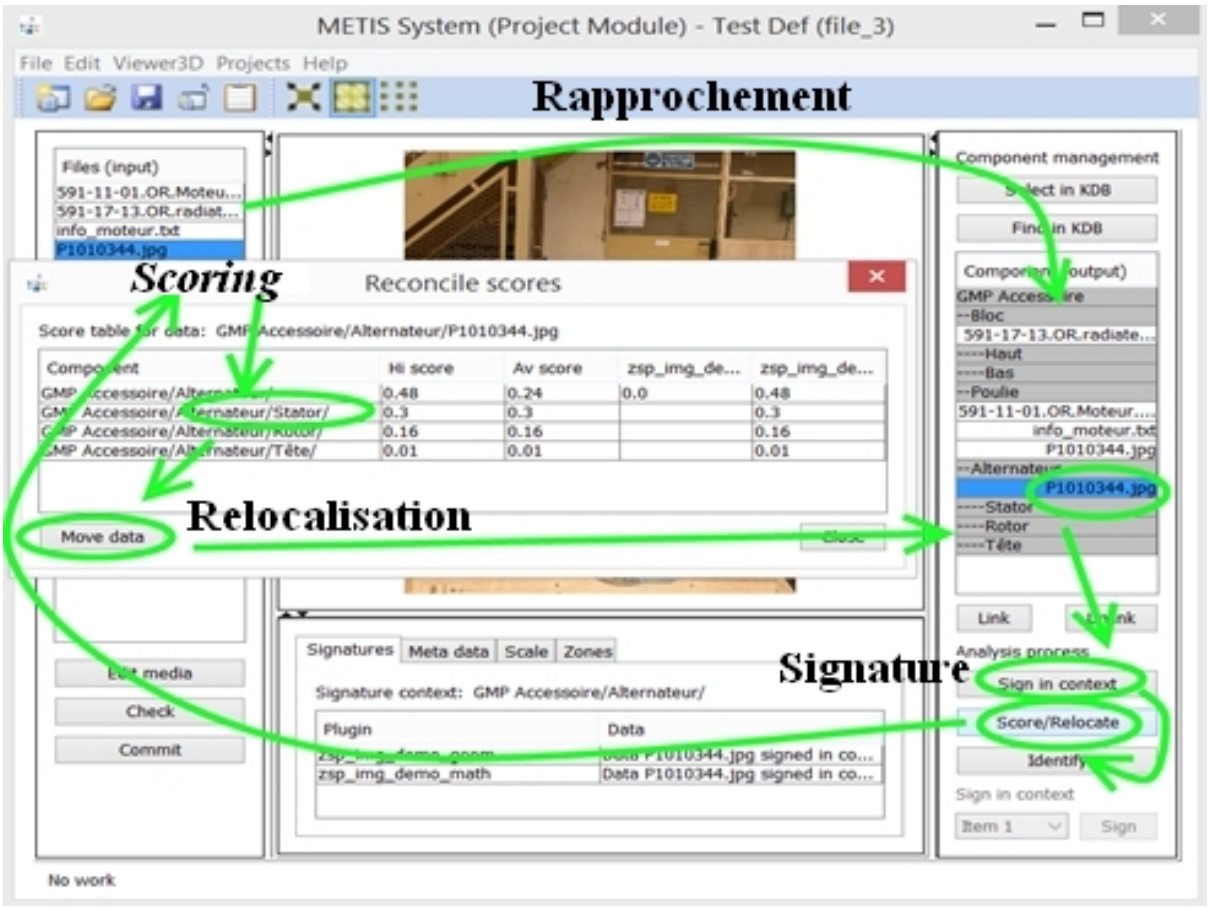


Figure C.8 – Étapes de signature et de scoring dans METIS (développement de l'interface réalisé par DeltaCAD).

Une fois que les composants ont été reconnus dans la donnée d'entrée, il est nécessaire de leur associer les composants correspondants de la base. Ceci est décrit dans la figure C.9.

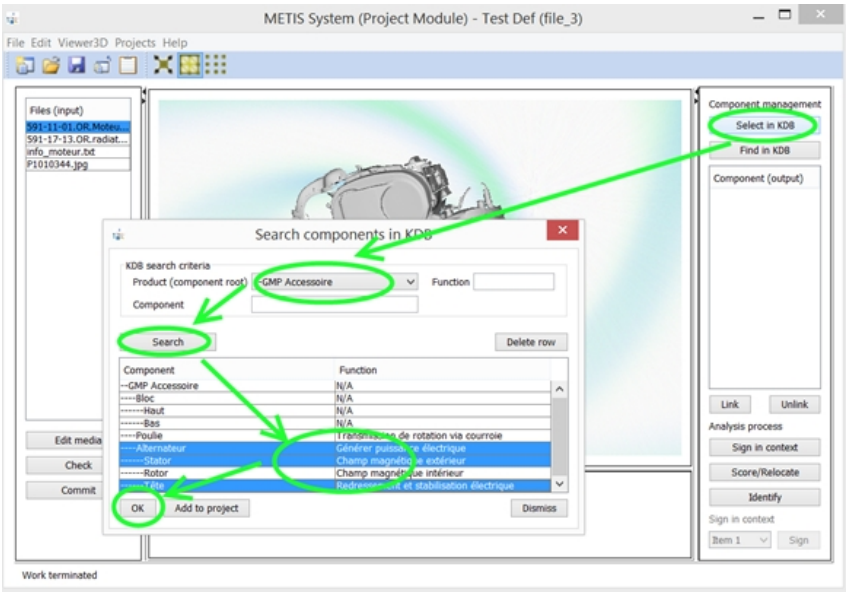


Figure C.9 – Mise en correspondance des résultats obtenus avec les composants en base.

Identification, évaluation des paramètres

Cette dernière étape, nommée “identification”, consiste à instancier les modèles CAO paramétrés en fonctions des valeurs extraites dans les données d’entrée. Nous pouvons l’observer dans la figure C.10.

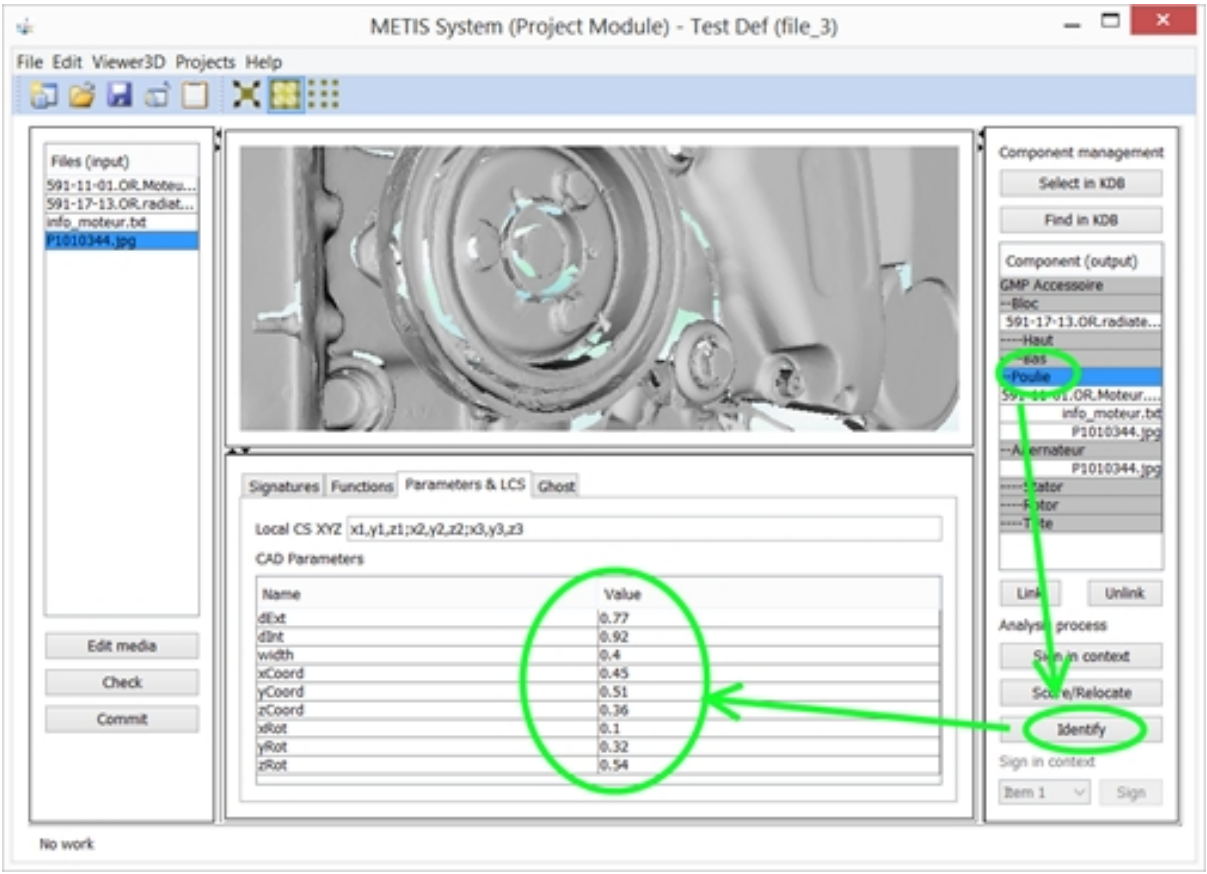


Figure C.10 – Identification des modèles CAO paramétrés et évaluation des paramètres (développement de l’interface réalisé par DeltaCAD).

Autres fonctions : sélection et zone

D’autres fonctions ont été développées telles que la sélection et le “zonage”. Ces fonctions permette de “rogner” les données et ainsi garder uniquement la partie à signer par exemple. Un exemple avec une poulie à l’intérieur d’un nuage de points est donné dans la figure C.11. Les points de cette poulie sont extraits du reste de la donnée.

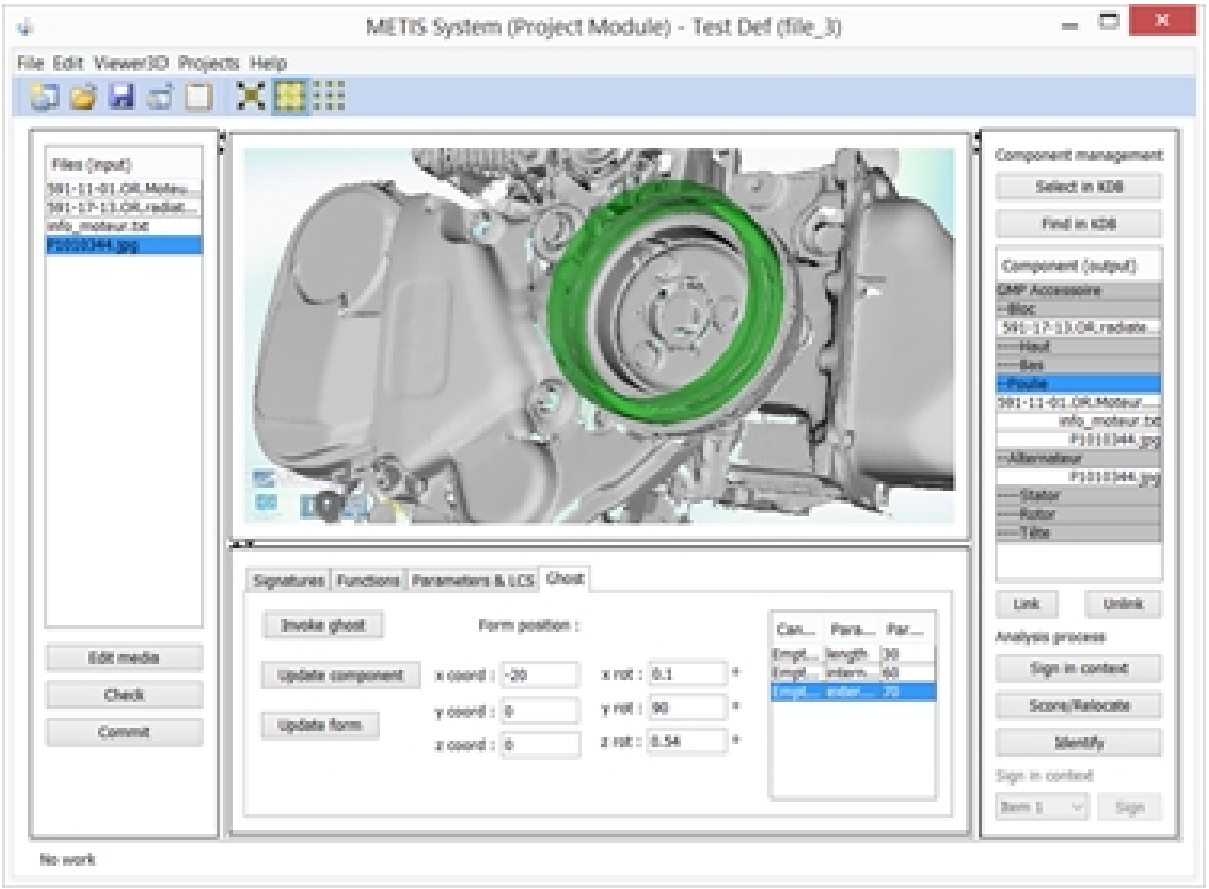


Figure C.11 – Sélection d'une zone dans un nuage de points dense (développement de l'interface réalisé par DeltaCAD).

Bibliographie

[Abella *et al.*, 1994] Abella, R. J., Daschbach, J. M. et McNichols, R. J. (1994). Reverse engineering industrial applications. *Computers and Industrial Engineering*, 26(2):381 – 385.

[Ali, 2015] Ali, S. (2015). *La rétro-conception de composants mécaniques par une approche : Concevoir pour Fabriquer*. Thèse de doctorat, Université de Technologie de Troyes, France.

[Amadori *et al.*, 2012] Amadori, K., Tarkian, M., Ölvander, J. et Krus, P. (2012). Flexible and robust CAD models for design automation. *Advanced Engineering Informatics*, 26(2):180–195.

[Ankerst *et al.*, 1999] Ankerst, M., Kastenmüller, G., Kriegel, H.-P. et Seidl, T. (1999). 3D Shape Histograms for Similarity Search and Classification in Spatial Databases. *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, (July):207–226.

[Army Research Office, 2003] Army Research Office (2003). Virtual Parts Engineering Research Initiative (VPERI). Rapport technique June, Arizona State University, Tempe.

[Ashby, 2005] Ashby, M. F. (2005). Materials selection in mechanical design. *MRS Bull*, 30(12):994–997.

[Attene *et al.*, 2006] Attene, M., Falcidieno, B. et Spagnuolo, M. (2006). Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193.

[Bagci, 2009] Bagci, E. (2009). Reverse engineering applications for recovery of broken or worn parts and re-manufacturing : Three case studies. *Advances in Engineering Software*, 40(6):407 – 418.

[Bärecke, 2009] Bärecke, T. (2009). *Isomorphisme inexact de graphes par optimisation évolutionnaire*. Thèse de doctorat, Université Pierre Marie Curie, Paris IV.

[Basri *et al.*, 1998] Basri, R., Costa, L., Geiger, D. et Jacobs, D. (1998). Determining the similarity of deformable shapes. *Vision Research*, 38(15-16):2365–2385.

[Bénière, 2012] Bénière, R. (2012). *Reconstruction d'un modèle B-Rep à partir d'un maillage 3D*. Thèse de doctorat, Université de Montpellier II, France.

[Benko *et al.*, 2001] Benko, P., Martin, R. R. et Varady, T. (2001). Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33(11):839–851.

[Blessing et Chakrabarti, 2009] Blessing, L. T. et Chakrabarti, A. (2009). *DRM, a Design Research Methodology*. Numéro 1. Springer London, London.

[Bruneau *et al.*, 2013] Bruneau, M., Durupt, A., Roucoules, L., Pernot, J.-P. et Eynard, B. (2013). Towards new processes to reverse engineering digital mock-ups from a set of heterogeneous data. *In Proceedings of the Ingegraph- ADM- AIP Primeca Conference*, pages 2–6, Madrid, Espagne.

[Bruneau *et al.*, 2014] Bruneau, M., Durupt, A., Roucoules, L., Pernot, J.-P. et Rowson, H. (2014). A methodology of Reverse Engineering for large mechanical assemblies products from heterogeneous data. *In* Horvath, I. et Rusak, Z., éditeurs : *Proceedings of TMCE 2014*, numéro 2, pages 1307–1318, Budapest, Hongrie.

[Bruneau *et al.*, 2016] Bruneau, M., Vallet, L., Durupt, A., Roucoules, L. et Pernot, J.-P. (2016). A three-levels signature by graph for reverse engineering of mechanical assemblies. *In* Horvath, I. et Rusak, Z., éditeurs : *Futurs Proceedings of TMCE 2016*, Aix-en-Provence, France.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–98.

[CGAL Project, 2015] CGAL Project, T. (2015). *CGAL : User and Reference Manual*. CGAL Editorial Board, 4.6.2 édition.

[Chang et Park, 2001] Chang, I. S. et Park, R.-H. (2001). Segmentation based on fusion of range and intensity images using robust trimmed methods. *Pattern Recognition*, 34(10):1951–1962.

[Chang, 2011] Chang, K.-H. (2011). 3D Shape Engineering and Design Parameterization. *Computer-Aided Design and Applications*, 8(5):681–692.

[Chaperon, 2002] Chaperon, T. (2002). *Segmentation de nuage de points 3D pour la modélisation automatique d'environnements industriels numérisés*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, France.

[Chapman et Pinfold, 1999] Chapman, C. B. et Pinfold, M. (1999). Design engineering - a need to rethink the solution using knowledge based engineering. *Knowledge-Based Systems*, 12:257–267.

[Chen *et al.*, 2009] Chen, X., Golovinskiy, A. et Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3).

[Collomb, 1980] Collomb, G. (1980). Estimation de la régression par la méthode des k points les plus proches avec noyau : quelques propriétés de convergence ponctuelle. *In* Raoult, J.-P., éditeur : *Statistique non Paramétrique Asymptotique*, volume 821 de *Lecture Notes in Mathematics*, pages 159–175. Springer Berlin Heidelberg.

[Colombo *et al.*, 2010] Colombo, G., Filippi, S., Rizzi, C. et Rotini, F. (2010). A new design paradigm for the development of custom-fit soft sockets for lower limb prostheses. *Computers in Industry*, 61(6):513 – 523. Soft Products Development.

[Coma *et al.*, 2003] Coma, O., Mascle, C. et Véron, P. (2003). Geometric and form feature recognition tools applied to a design for assembly methodology. *Computer-Aided Design*, 35(13):1193 – 1210.

[Conte *et al.*, 2004] Conte, D., Foggia, P., Sansone, C. et Vento, M. (2004). Thirty years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298.

[Cyr et Kimia, 2003] Cyr, C. M. et Kimia, B. B. (2003). 3D object recognition using shape similiarity-based aspect graph. *In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 254–261, Vancouver, BC. IEEE Comput. Soc.

[De Luca, 2006] De Luca, L. (2006). Relevé et multi-représentations du patrimoine architectural. *MIA-journal*, (1):1–12.

[De Luca *et al.*, 2006] De Luca, L., Veron, P. et Florenzano, M. (2006). Reverse engineering of architectural buildings based on a hybrid modeling approach. *Computers & Graphics*, 30(2):160–176.

[Dekhtiar *et al.*, 2016] Dekhtiar, J., Durupt, A., Bricogne, M., Kiritsis, D., Rowson, H. et Eynard, B. (2016). Toward an extensive data integration to address reverse engineering issues. Non publié.

[Delest *et al.*, 2007] Delest, S., Boné, R. et Cardot, H. (2007). Etat de l'art de la segmentation de maillage 3D par patchs surfaciques. *In GTMG '07 : Groupe de Travail en Modélisation Géométrique*, pages 171–185, Valenciennes (France).

[Deveau, 2006] Deveau, M. (2006). *Utilisation conjointe de données image et laser pour la segmentation et la modélisation 3D*. Thèse de doctorat, Université René Descartes - Paris 5, France.

[Developpez, 2000] Developpez, L. (2000). Classes de traits et de politiques en c++. Consulté le 03/06/2015.

[Dúbravčík et Štefan Kender, 2012] Dúbravčík, M. et Štefan Kender (2012). Application of reverse engineering techniques in mechanics system services. *Procedia Engineering*, 48(0):96 – 104. Modelling of Mechanical and Mechatronics Systems.

[Durupt, 2010] Durupt, A. (2010). *Définition d'un processus de rétro-conception de produit par intégration des connaissances de son style de vie*. Thèse de doctorat, Université de Technologie de Troyes, France.

[Durupt *et al.*, 2009] Durupt, A., Remy, S., Ducellier, G. et Guyot, E. (2009). A new reverse engineering process, the combination between the knowledge extraction and the geometrical recognition techniques. *2009 International Conference on Computers & Industrial Engineering*, pages 1367–1372.

[El-Mehalawi et Miller, 2003a] El-Mehalawi, M. et Miller, R. A. (2003a). A database system of mechanical components based on geometric and topological similarity. Part I : Representation. *CAD Computer Aided Design*, 35(1):83–94.

[El-Mehalawi et Miller, 2003b] El-Mehalawi, M. et Miller, R. A. (2003b). A database system of mechanical components based on geometric and topological similarity. Part II : indexing, retrieval, matching and similarity assesment. *CAD Computer Aided Design*, 35(1):95 – 105.

[Fan *et al.*, 1987] Fan, T.-J., Medioni, G. et Nevatia, R. (1987). Segmented descriptions of 3-D surfaces. *IEEE Journal on Robotics and Automation*, 3(6):527–538.

[FaroLaser, 2012] FaroLaser (2012). Forensics and crime scene analysis - faro solutions. http://www.faro.com/measurement-solutions/applications/crime-scene-analysis. Consulté le 06/05/2015.

[Fischler et Bolles, 1981] Fischler, M. et Bolles, R. C. (1981). Random Sample Consensus : A Paradigm for Model Fitting with Applicatlons to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381 – 395.

[Fisher, 2004] Fisher, R. B. (2004). Applying knowledge to reverse engineering problems. *Computer-Aided Design*, 36(6):501–510.

[Funkhouser *et al.*, 2004] Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S. et Dobkin, D. (2004). Modeling by example. *ACM Transactions on Graphics*, 23(3):652.

[Gélineau, 2013] Gélineau, S. (2013). Société ctec-3. http://www.ctec3d.com. Consulté le 05/05/2015.

[Hendel, 2008] Hendel, T. (2008). File Exchange MATLAB Central. Consulté le 03/11/2014.

[Herlem *et al.*, 2012] Herlem, G., Adragna, P.-a., Ducellier, G. et Durupt, A. (2012). DMU Maturity Management as an Extension of the Core Product Model. *Proceedings of the International Conference on Product Lifecycle Management*, Volume 388:192–201.

[Hichri *et al.*, 2013] Hichri, N., Chiara, S., De Luca, L. et Véron, P. (2013). Review of the "AS-BUILT BIM" approaches. *In 3D-ARCH 2013*, volume XL-5/W1, pages 107–112, Trento, Italy.

[Hough, 1962] Hough, P. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.

[Jaulent, 1992] Jaulent, P. (1992). *Génie logiciel : les méthodes SADT, SA, EA, SA-RT, SYS-PO, OOD, HOOD*. Armand Colin.

[Jennings, 2011] Jennings, A. (2011). File Exchange MATLAB Central. Consulté le 03/11/2014.

[Johnson, 2008] Johnson, E. (2008). File Exchange MATLAB Central. Consulté le 03/06/2014.

[Jusufi, 2013] Jusufi, I. (2013). *Multivariate Networks : Visualization and Interaction Techniques*. Thèse de doctorat, Linnæus University (Suède).

[Kermad, 1997] Kermad, C. (1997). *Segmentation d'image : recherche d'une mise en oeuvre automatique par coopération de méthodes*. Thèse de doctorat, Université de Rennes 1, France.

[Korsawe, 2008] Korsawe, J. (2008). File Exchange MATLAB Central. Consulté le 03/06/2014.

[Laroche *et al.*, 2008] Laroche, F., Bernard, A. et Cotte, M. (2008). Advanced industrial archaeology : A new reverse-engineering process for contextualising and digitising ancient technical objects. *Virtual and Physical Prototyping*, 3(2):105–122.

[Lartigue *et al.*, 2002] Lartigue, C., Contri, A. et Bourdet, P. (2002). Digitised point quality in relation with point exploitation. *Measurement : Journal of the International Measurement Confederation*, 32(3):193–203.

[Leventon, 2015] Leventon, M. (2015). Michael Leventon's Home Page. Consulté le 03/07/2015.

[Lévy *et al.*, 2002] Lévy, B., Petitjean, S., Ray, N. et Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. *In* ACM, éditeur : *ACM SIGGRAPH conference proceedings*.

[Leygue, 2013] Leygue, A. (2013). File Exchange MATLAB Central. Consulté le 03/11/2014.

[Liu *et al.*, 2003] Liu, G., Wong, Y., Zhang, Y. et Loh, H. (2003). Error-based segmentation of cloud data for direct rapid prototyping. *Computer-Aided Design*, 35(7):633–645.

[Liu et Siegwart, 2013] Liu, M. et Siegwart, R. (2013). Information theory based validation for point-cloud segmentation aided by tensor voting. *In Information and Automation (ICIA), 2013 IEEE International Conference on*, pages 168–173.

[Liu et Ma, 1999] Liu, S. et Ma, W. (1999). Seed-growing segmentation of 3-D surfaces from CT-contour data. *Computer-Aided Design*, 31(8):517–536.

[Lovett *et al.*, 2000] Lovett, P., Ingram, A. et Bancroft, C. (2000). Knowledge-based engineering for {SMEs} — a methodology. *Journal of Materials Processing Technology*, 107(1–3):384 – 389.

[Martin *et al.*, 2001] Martin, D., Fowlkes, C., Tal, D. et Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *In Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

[Milroy *et al.*, 1997] Milroy, M., Bradley, C. et Vickers, G. (1997). Segmentation of a wrap-around model using an active contour. *Computer-Aided Design*, 29(4):299–320.

[Montero et Bribiesca, 2009] Montero, R. S. et Bribiesca, E. (2009). State of the Art of Compactness and Circularity Measures. *International Mathematical Forum*, 4(27):1305–1335.

[Motavalli, 1998] Motavalli, S. (1998). Review of Reverse Engineering Approaches. *Computers & Industrial Engineering*, 35(98):25–28.

[Nüchter et Hertzberg, 2008] Nüchter, A. et Hertzberg, J. (2008). Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926.

[Oldham *et al.*, 1998] Oldham, K., Kneebone, S., Callot, M., Murton, A. et Brimble, R. (1998). MOKA - A methodology and tools oriented to knowledge based applications. *In Proceedings of the Conference on Integration in Manufacturing*, pages 198–207, Göteborg, Sweden.

[Osada *et al.*, 2002] Osada, R., Funkhouser, T., Chazelle, B. et Dobkin, D. (2002). Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832.

[Othman *et al.*, 2009] Othman, Z., Rafiq, M. et Kadir, A. (2009). Comparison of Canny and Sobel Edge Detection in MRI Images. *Computer Science, Biomechanics \& Tissue Engineering Group, and Information System*, pages 133–136.

[Ouamer-Ali *et al.*, 2014] Ouamer-Ali, M.-I., Laroche, F., Bernard, A. et Remy, S. (2014). Toward a methodological knowledge based approach for partial automation of reverse engineering. *Procedia {CIRP}*, 21(0):270 – 275. 24th {CIRP} Design Conference.

[Pal *et al.*, 2006] Pal, D. K., Ravi, B., Bhargava, L. S. et Chandrasekhar, U. (2006). Computer-aided reverse engineering for rapid replacement of parts : A Case Study. *Defence Science Journal*, 56(2):225–238.

[Parker *et al.*, 2001] Parker, K. J., Marjatta, S., Sofia, T. et Jose, T. P. (2001). System and methode for 4D reconstruction and visualization.

[Petre *et al.*, 2010] Petre, R.-D., Zaharia, T. et Prêteux, F. (2010). An experimental evaluation of view-based 2d/3d indexing methods. *In Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*, pages 000924–000928.

[Picard, 2009] Picard, J. (2009). En route vers le zéro prototype. *L'Usine Nouvelle*, (n°3137).

[Quinsat et Lartigue, 2015] Quinsat, Y. et Lartigue, C. (2015). Filling holes in digitized point cloud using a morphing-based approach to preserve volume characteristics. *The International Journal of Advanced Manufacturing Technology*, 81(1-4):411–421.

[Raja et Fernandes, 2007] Raja, V. et Fernandes, K. J. (2007). *Reverse Engineering : An Industrial Perspective*. Springer Publishing Company, Incorporated, 1st édition.

[Rathore et Jain, 2014] Rathore, N. et Jain, P. K. (2014). Reverse Engineering Applications in Manufacturing Industries : an Overview. *DAAAM International Scientific Book 2014*, pages 567–576.

[Reeb, 1946] Reeb, G. (1946). Sur les points singuliers d'une forme de pfaff completement intégrable ou d'une fonction numérique. *CR Acad. Sci. Paris*, 222:847–849.

[Rhinoceros3D, 2012] Rhinoceros3D (2012). Rhinoceros - nurbs. www.rhino3d.com/nurbs. Consulté le 27/05/2015.

[Rivera, 2009] Rivera, R. (2009). Boost c++ librairies. Consulté le 03/10/2015.

[Sánchez-Cruz et Bribiesca, 2003] Sánchez-Cruz, H. et Bribiesca, E. (2003). A method of optimum transformation of 3D objects used as a measure of shape dissimilarity. *Image and Vision Computing*, 21(12):1027–1036.

[Shamir, 2008] Shamir, A. (2008). A survey on Mesh Segmentation Techniques. *Computer Graphics Forum*, 27(6):1539–1556.

[Sobel, 1978] Sobel, I. (1978). Neighborhood coding of binary images for fast contour following and general binary array processing. *Computer Graphics and Image Processing*, 8(1):127–135.

[Spirkovska, 1993] Spirkovska, L. (1993). A Summary of Image Segmentation Techniques. Rapport technique June 1993, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California.

[Sri Madhava Raja *et al.*, 2014] Sri Madhava Raja, N., Rajinikanth, V. et Latha, K. (2014). Otsu based optimal multilevel image thresholding using firefly algorithm. *Modelling and Simulation in Engineering*, 2014:17.

[Sundar *et al.*, 2003] Sundar, H., Silver, D., Gagvani, N. et Dickinson, S. (2003). Skeleton based shape matching and retrieval. *In* IEEE, éditeur : *Shape Modeling International 2003*, pages 130–139.

[Sunil et Pande, 2008] Sunil, V. et Pande, S. (2008). Automatic recognition of features from freeform surface {CAD} models. *Computer-Aided Design*, 40(4):502 – 517.

[Tangelder et Veltkamp, 2003] Tangelder, J. et Veltkamp, R. C. (2003). Polyhedral model retrieval using weighted point sets. *2003 Shape Modeling International.*

[Tangelder et Veltkamp, 2007] Tangelder, J. W. H. et Veltkamp, R. C. (2007). A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471.

[Technopedia, 2015] Technopedia (2015). What is deep learning ? Consulté le 08/01/2016.

[Thilmany, 2012] Thilmany, J. (2012). The rise of Reverse Engineering. Consulté le 25/09/2015.

[Thompson *et al.*, 1999] Thompson, W., Owen, J., de St.Germain, H., Stark, S. J. et Henderson, T. (1999). Feature-based reverse engineering of mechanical parts. *Robotics and Automation, IEEE Transactions on*, 15(1):57–66.

[Tierny *et al.*, 2006] Tierny, J., Vandeborre, J.-p. et Daoudi, M. (2006). Graphes de Reeb de Haut Niveau de Maillages Polygonaux 3D. *In COmpression et REprésentation des Signaux Audiovisuels (CORE-SA'06)*, pages 172–177, Caen (France).

[Tung et Schmitt, 2005] Tung, T. et Schmitt, F. (2005). The augmented multiresolution reeb graph approach for content-based retrieval of 3d shapes. *International Journal of Shape Modeling*, 11(01):91–120.

[Várady *et al.*, 1997] Várady, T., Martin, R. R. et Cox, J. (1997). Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268.

[Veltkamp, 2001] Veltkamp, R. (2001). Shape matching : similarity measures and algorithms. *In Shape Modeling and Applications, SMI 2001 International Conference on*, pages 188–197.

[Veltkamp et Hagedoorn, 2001] Veltkamp, R. C. et Hagedoorn, M. (2001). State of the art in shape matching. *In* Lew, M. S., éditeur : *Principles of Visual Information Retrieval*, chapitre State of the art in shape matching, pages 87—-119. Springer-Verlag.

[Visionair, 2015] Visionair (2015). Digital Shape Workbench. Consulté le 03/07/2015.

[Vo *et al.*, 2015] Vo, A.-V., Truong-Hong, L., Laefer, D. F. et Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:88–100.

[Wang *et al.*, 2012] Wang, J., Gu, D., Yu, Z., Tan, C. et Zhou, L. (2012). A framework for 3d model reconstruction in reverse engineering. *Computers and Industrial Engineering*, 63(4):1189 – 1200.

[Wang, 2013] Wang, W. (2013). Application of Reverse Engineering in Manufacturing Industry. *In EASTEC 2013, SME*, pages 1–7, West Springfield, MA (USA).

[Wani et Batchelor, 1994] Wani, M. et Batchelor, B. (1994). Edge-region-based segmentation of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):314–319.

[Wikipédia, 2015] Wikipédia (2015). Moteur d'inférence. Consulté le 03/10/2015.

[Wilson et Zhu, 2008] Wilson, R. C. et Zhu, P. (2008). A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41:2833–2841.

[Woo *et al.*, 2002] Woo, H., Kang, E., Wang, S. et Lee, K. H. (2002). A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture*, 42(2):167–178.

[Xiao *et al.*, 2011] Xiao, D., Lin, H., Xian, C. et Gao, S. (2011). CAD mesh model segmentation by clustering. *Computers & Graphics*, 35(3):685–691.

[Yang et Lee, 1999] Yang, M. et Lee, E. (1999). Segmentation of measured point data using a parametric quadric surface approximation. *Computer-Aided Design*, 31(7):449–457.

[Yang *et al.*, 2008] Yang, T., Liu, B. et Zhang, H. (2008). 3d model retrieval based on exact visual similarity. *In Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pages 1556–1560.

[Yau *et al.*, 1993] Yau, H., Haque, S. et Menq, C. (1993). Reverse engineering in the design of engine intake and exhausts ports. *ASME PROD ENG DIV PUBL PED*, 64:139–148.

[Zhan *et al.*, 2010] Zhan, Q., Yu, L. et Liang, Y. (2010). A point cloud segmentation method based on vector estimation and color clustering. *In 2nd International Conference on Information Science and Engineering, ICISE2010*, pages 3463–3466, Hangzhou, China.