Sélection et amélioration de nuages de points 3D

Adrien Hamelin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Image, Information et Hypermédia

Présentée et soutenue par :

M. ADRIEN HAMELIN

le mercredi 16 décembre 2015

Titre :

SELECTION ET AMELIORATION DE NUAGES DE POINTS 3D

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

M. EMMANUEL DUBOIS

M. REMI CABANAC

Rapporteurs :

M. JEAN-PHILIPPE PERNOT, ENSAM - ARTS ET METIERS PARISTECH

Mme NADINE COUTURE, ESTIA BIDART

Membre(s) du jury :

M. JEAN-PHILIPPE PERNOT, ENSAM - ARTS ET METIERS PARISTECH, Président

M. EMMANUEL DUBOIS, UNIVERSITE TOULOUSE 3, Membre

Mme MINICA HOURY-PANCHETTI, UNIVERSITE TOULOUSE 3, Membre

Résumé

Le domaine de l'informatique 3D est vaste, contenant de nombreuses directions de recherche possible. Une de ces directions est la numérisation d'objets réels, via des scanners. De nombreux objectifs peuvent requérir cela, de l'affichage et de l'interaction avec le modèle dans un environnement 3D comme un jeu vidéo à l'analyse dudit modèle pour de la rétro-ingénierie, comme l'analyse de l'évolution de l'objet dans le temps pour de la maintenance. Le problème est alors d'être capable de traiter les modèles ainsi acquis.

Cette thèse se focalise plus particulièrement sur les modèles de bâtiments acquis à l'aide d'un scanner laser, résultant en un nuage de points 3D. Le processus d'acquisition n'étant pas parfait, le modèle obtenu comporte des défauts comme des trous ou du bruit.

Notre première contribution est alors de pouvoir sélectionner une partie de ce nuage de points, dans le but par exemple de cibler des traitements. Le principe est de faire une sélection par étape en créant un volume formé d'une succession de contours de forme libre. Des tests expérimentaux ont montré que cette technique est efficace et préférée par rapport aux deux autres techniques de sélection testées.

Notre deuxième contribution est un processus visant à corriger les défauts dus à l'acquisition du modèle. Pour cela, le processus se base sur d'autres sources contenant des informations sur l'objet représenté pour détecter un ensemble de zones contenant des problèmes. Il essaie ensuite autant que possible de prendre automatiquement des décisions pour la correction, mais laisse toujours le choix final à un utilisateur. Les résultats montrent que le modèle obtenu est d'une bien meilleure qualité visuelle.

Abstract

The computer science field is vast, containing numerous areas of possible research focus. One of those areas is the transposition of real objects in 3D worlds, using scanners. A variety of goals can need such a step, from displaying and interacting with the resulting model in a 3D environment such as a video game to analyzing that model for retro-engineering, such as analyzing the evolution in time of the object for maintenance. The problem then becomes to be capable to treat the models thus acquired.

This thesis focuses more particularly on models of building, acquired with a laser scanner, giving a 3D point cloud. The acquisition process being not perfect, the resulting model contains defects such as holes or noise.

Our first contribution is then to enable a user to select a part of the cloud, for example to target treatments. The principle is to perform the selection in several steps by creating a volume composed of successive free-formed outlines. Experimental tests showed that this technique is effective and preferred to the two other compared selection techniques.

Our second contribution is a process aiming at correcting the defects caused by the acquisition. To that end, the process bases itself on other sources having information on the represented object to detect a set of areas containing problems. It tries then as much as possible to take automatic decisions to correct the problems, but always leaves the final choice to a user. The experiment showed the the resulting model looks much better.

Remerciements

J'aimerais tout d'abord remercier mes encadrants, qui ont été présents pour moi et m'ont guidés tout au long de cette thèse. Émmanuel Dubois et Rémi Cabanac, mes directeurs de thèse, ainsi que Minica Houry Panchetti et Patrice Torguet. J'aimerais également remercier les équipes de recherche qui m'ont accueillie, Vortex et Elipse. Les chercheurs de ces équipes pour leur aide quand j'en avais besoin, mais également les doctorants qui ont rendu cette entreprise agréable, tout particulièrement au sein de l'équipe Elipse. J'aimerais enfin remercier ma famille, qui m'a soutenu tout du long, et les participants à mes évaluations expérimentales, qui n'étaient pas faciles à digérer. Merci beaucoup à tous, ce fut trois années que je ne regrette pas et je n'y serais pas arrivé sans vous.

Table des matières

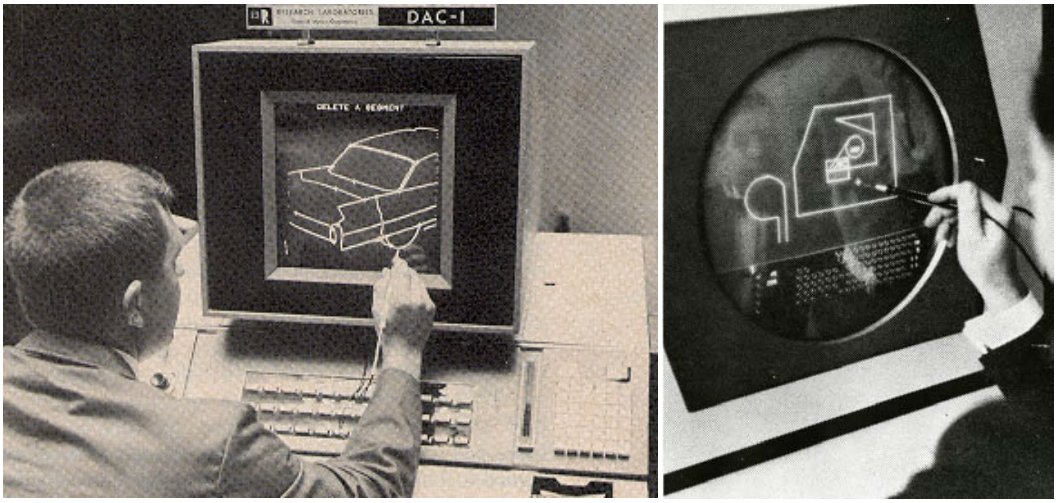
Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Histoire du rendu graphique [1]



(a) La machine DAC-1, photo de http://www.turkcadcam.net/

(b) Sketchpad, photo de http://www.computerhistory.org/

Figure 1.1 – Les premiers rendus graphiques

La première machine destinée aux rendus graphiques fût créée en 1959. Il s'agissait de la machine DAC-1 (*Design Augmented by Computers*, figure 1.1a), dévoilée au public en 1964. Elle permettait à un utilisateur de rentrer une description 3D d'une voiture et de la visualiser depuis différents points de vue.

Mais s'il fallait parler d'un pionnier du domaine, cette personne serait Ivan Sutherland. En 1961, alors étudiant au MIT, il créa Sketchpad (figure 1.1b), un logiciel de dessin qui permettait de dessiner des formes simples directement sur l'écran et de les sauvegarder. Ce logiciel fut la base de nombreux autres logiciels dédiés au dessin, avec une influence encore visible aujourd'hui. Par exemple, le

1. Source : http://www.danielsevo.com/

logiciel propose de dessiner des formes comme des carrés et pas uniquement des lignes, ce qui permet d'avoir ces formes parfaites au lieux d'un ensemble de lignes faîtes à la main.

De nombreuses entreprises se sont ensuite lancées dans le rendu graphique, dont IBM qui lança en 1964 le *IBM 2250 graphics terminal*, le premier ordinateur dédié aux rendus graphiques disponible commercialement.

FIGURE 1.2 – Aperçu de *A Computer Animated Hand*, de http://pixartimes.com/

De nombreuses avancées dans le domaine furent réalisées dans les années suivantes. De l'utilisation et l'affichage de polygones à l'animation, ces avancées permirent à Edwin Catmull, étudiant de Ivan Sutherland, de créer en 1972 une petite vidéo nommée *A Computer Animated Hand* (dont un aperçu est visible figure 1.2), qui montrait l'animation d'une main composée de 350 polygones, créée à l'aide d'une acquisition d'un modèle de main réel. Cette vidéo fut par la suite reprise en 1976 dans le film *Futureworld*, qui devint ainsi le premier film à utiliser un rendu graphique en 3D.

Le domaine continua ensuite d'évoluer, principalement à l'université de l'Utah qui comportait Ivan Sutherland dans ses membres. L'université attira de nombreuses personnes, parmi lesquelles Edwin Catmull qui fonda Pixar ou John Warnock qui fonda Adobe Systems. Le groupement SIGGRAPH fût créé en 1973 et a beaucoup aidé au développement du domaine, en étant jusqu'à maintenant l'organisateur des conférences parmi les plus grandes.

Dans les années 80, avec la démocratisation de l'ordinateur dans les foyers, le

nombre de développeurs graphique augmenta grandement. Les jeux vidéos commencèrent à trouver leur public et exposa le domaine du rendu graphique aux plus jeunes. La carte graphique trouva également son origine à cette époque. Les années 90 arrivèrent alors et virent l'émergence de la modélisation 3D à grande échelle, et de son utilisation dans les différents programmes avec les nouveaux standards, DirectX et OpenGl. Le premier film d'animation vit également le jour : *Toy Story* [Pixar 1995]. Enfin, les années 2000 furent les témoins de la nette amélioration de la qualité graphique des rendus, en affichant beaucoup de détails et de réalisme. La 3D devint couramment utilisée dans les films et les séries. Aujourd'hui, la qualité des rendus graphiques est suffisante pour la majorité des applications.

Cet historique montre bien que dès le commencement du rendu graphique par ordinateur, à la fois les modèles 3D mais aussi l'interaction avec un utilisateur étaient au centre des préoccupations. Dès le premier exemple montré, avec la DAC-1, les modèles gérés étaient en 3D, et manipulables directement avec un stylet sur un écran. Aujourd'hui, la complexité des modèles n'a fait qu'augmenter, ainsi que les capacités des ordinateurs, et des traitements de plus en plus avancés voient le jour. Cette thèse détaille ma contribution à ces avancées.

1.2 Contexte

1.2.1 Général

De nombreuses représentations existent pour figurer un modèle 3D. Ces représentations ont pour origine des sources différentes et viennent avec leur lot d'avantages et d'inconvénients. Cependant certaines limitations sont trop importantes et ne permettent pas d'obtenir des résultats satisfaisants. Dans le contexte de l'acquisition 3D de bâtiments réels en particulier, les techniques qui permettent de transposer ces bâtiments en modèles comportent d'importantes limitations. L'une d'elles, peut être la plus importante, est le manque d'information ; bien que la résolution du modèle soit importante, des parties sont tout simplement absentes du résultat. C'est par exemple le cas du dessus d'un toit si le scanner n'a été placé qu'au niveau du sol. Une autre est le bruit : des points qui ne représentent pas la surface.

Ces problèmes sont importants pour de nombreuses raisons ; pour la visualisation tout d'abord, cela limite le champs des possibles, mais ce manque d'information est aussi néfaste pour les autres traitements possibles (dont des exemples ont été donnés dans le précédent chapitre). Des techniques ont alors été inventées pour essayer de combler ces manques, comme le remplissage de trous. Mais à moins de connaître l'objet à reconstruire, ces traitements ne font qu'essayer de deviner la réalité pour la reconstruire. Une information qui n'existe pas dans le modèle ne pourra jamais être retrouvée de façon certaine, et les résultats dépendront alors toujours des suppositions plus ou moins intelligentes faites a priori sur le modèle à réparer. Ces algorithmes ont alors besoin de se spécialiser pour obtenir des résultats satisfaisants, mais réduisent de fait leur ouverture à un large champs de modèles reconstructibles. Cependant, si l'information qui manque n'existe pas dans le modèle acquis, cela ne signifie pas nécessairement que cette information n'existe pas du tout.

En effet, les objets que nous pouvons retrouver dans la vie de tous les jours ne sont pas créés à partir de rien. Pour un bâtiment par exemple, des plans existent souvent qui permettent de le construire. Et l'informatique est un outil formidable pour l'aide à la conception. De nombreux bâtiments sont ainsi créés virtuellement, en 3D, avant d'entamer la construction elle-même. Cela permet, par exemple, à une personne souhaitant construire une maison de se représenter plus facilement le résultat, voir même de marcher dedans à l'aide de technologies immersives, avant même la pose de la première pierre.

Ces modèles n'enlèvent pas l'intérêt de numériser les objets après construction ou fabrication. En effet, ces modèles représentent l'objet tel que le concepteur souhaiterait qu'il soit. Mais de la même manière que pour les scanners, les procédés de fabrication ne sont pas parfaits. De plus l'objet va évoluer dans le temps, s'altérer. Il est donc intéressant de pouvoir comparer des acquisitions récentes face aux modèles de conception ou à d'autres acquisitions un peu plus anciennes.

Un autre exemple d'application pourrait être dans le cas d'une agence de location de voiture. L'agence pourrait, à chaque retour, numériser la voiture rendue et comparer le résultat à la précédente acquisition, pour détecter si des dommages ont été créés. L'exemple de la figure 1.3 montre bien le principe.

Ces modèles possèdent des informations sur l'objet représenté, informations

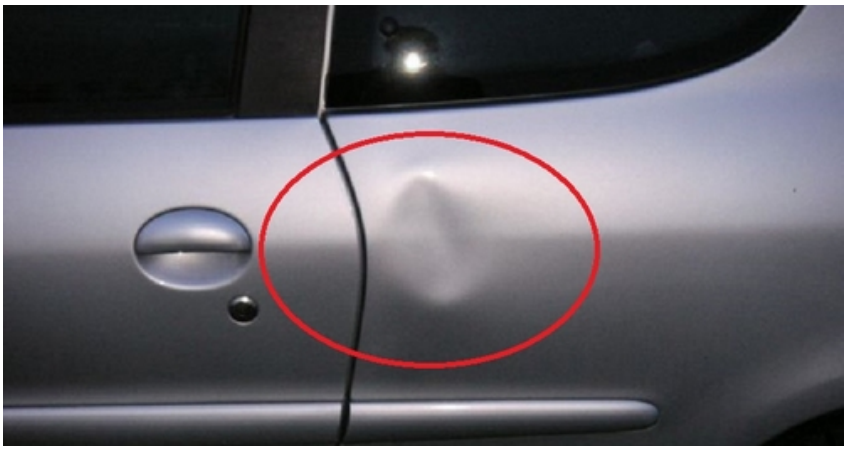


FIGURE 1.3 – Voiture cabossée, photo de http://www.letribunaldunet.fr/

qui peuvent permettre de compléter les manques d'autres modèles. L'idée proposée par cette thèse est alors exactement cela : réparer et compléter un modèle en combinant les informations de plusieurs sources et plusieurs représentations.

1.2.2 Projet

FIGURE 1.4 – Le Pic du Midi, photo de http://www.laroutedesorigines.fr/

Cette thèse et ses objectifs s'insèrent dans un projet concret. Cela commença quand l'université de Toulouse décida de numériser son patrimoine, pour plusieurs raisons. La première est que l'université possède un logiciel pour le suivi des bâtiments, mais que certains, trop vieux, n'avaient même pas de plans. L'ac-

quisition en 3D de ceux-ci permet d'en construire un. Une deuxième raison est pour la sauvegarde du patrimoine. Un modèle 3D permet, d'une part, d'en afficher la représentation dans des environnements 3D pour des buts publics mais également, en cas d'incidents comme un incendie, d'avoir une comparaison pour évaluer les dégâts et éventuellement de reconstruire à l'identique.

L'université a donc mandaté un cabinet qui a tout d'abord envoyé une équipe pour acquérir tous les bâtiments demandés en 3D, puis travailler à la main pour obtenir des modèles maillés, puisqu'il n'existe pas de méthode automatique fiable pour le faire.

Dans ce patrimoine sont inclus des bâtiments qui accueillent les étudiants, mais aussi des lieux de travail des chercheurs. C'est ainsi que les bâtiments au sommet du Pic du Midi, comprenant le Télescope Bernard Lyot (TBL), ont été scannés. Une photo du Pic du Midi est visible figure 1.4, et un aperçu du résultat de l'acquisition est montré sur la figure 1.5.

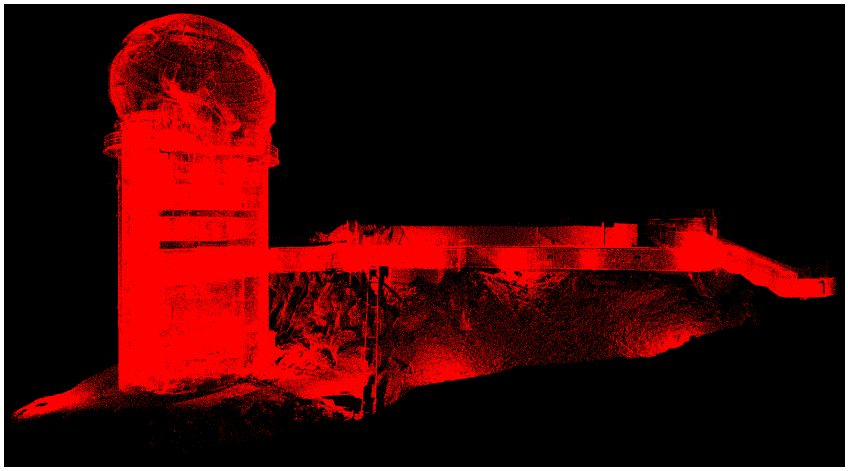


FIGURE 1.5 – Le Pic du Midi, en nuage de points

Le nuage de points a été acquis à l'aide d'un scanner laser. Le principe est d'envoyer une impulsion laser, qui va ensuite être réfléchie par la surface en train d'être scannée, et va être mesurée par un capteur présent sur le laser. Connaissant ainsi le temps que le laser a mis pour faire l'aller-retour, la distance peut être déduite. Cette impulsion est ensuite envoyée dans une direction légèrement différente, et le processus est répété. Le capteur va ainsi complètement tourner sur lui-même et acquérir toutes les distances des objets autour de lui. La précision de la capture va ainsi dépendre des capteurs utilisés, mais aussi du matériau de la

surface scannée et de la distance totale que l'impulsion laser doit parcourir.

L'acquisition complète se fait en plusieurs étapes. La première est de placer le scanner proche de l'objet a acquérir. Mais une seule position par rapport à l'objet ne suffit pas, car le laser n'est pas capable de traverser les surfaces. Il faut donc faire une succession de placements du scanner pour lui donner accès à toutes les parties de l'objet et faire une acquisition à chacun de ces placements. Cela permet également de prendre l'intérieur et l'extérieur d'un bâtiment par exemple.

La seconde étape est alors de récupérer toutes ces parties de modèle et de les combiner pour en faire une seule. Pour cela, il faudra avoir pris soin que les parties scannées se recoupent entre elles, afin d'avoir de l'information redondante. Pour aider, il est également généralement placé dans la scène (avant l'acquisition) des petits objets facilement reconnaissables. À l'aide de ces informations, il est possible de déterminer comment les acquisitions se situent les unes par rapport aux autres, et donc d'obtenir le modèle unique désiré. Le résultat est, dans notre cas, un modèle de 330 millions de points.

Par ailleurs, les responsables du TBL se sont interrogés sur l'utilisation de modèles 3D pour concevoir un nouvel outil pouvant améliorer les activités de tout le personnel travaillant sur ce site, les scientifiques de l'Observatoire Midi Pyrénées (OMP), les responsables de la maintenance ou encore les gens s'occupant du musée présent au sommet. C'est ainsi qu'un projet est né, avec des buts variés.

La première idée est d'utiliser le nuage de points ainsi acquis pour télé-opérer le télescope, avec une représentation 3D qui montrerait son état courant. Cela permettrait aussi de créer une simulation qui rendrait possible l'apprentissage des commandes pour les opérateurs en toute sécurité. Une deuxième idée concerne le musée. Cette représentation permet non seulement de faire découvrir le télescope au public, mais aussi d'interagir avec en montrant par exemple le trajet de la lumière au sein de celui-ci. Cet aspect de découverte par le public a déjà fait l'objet de recherches [Bergé *et al.* 2014]. D'autres idées, similaires à celles prévues au départ par l'université, concernent la maintenance.

Il y avait déjà eu plusieurs tentatives de créer un modèle 3D avec divers outils, mais sans donner suffisamment de satisfaction. Un bon indice de cela est que les modèles ainsi créés ne sont pas identiques.

Le projet a donc fourni plusieurs données qui ont servi de test à la thèse. La

première est le nuage de points du Pic du Midi. Ce nuage de point a résulté de l'assemblage de plusieurs acquisitions faites à l'aide d'un scanner laser placé à différents endroits. Il est donc relativement complet, et contient une grande partie du bâtiment, intérieur comme extérieur. Il représente la réalité au moment de l'acquisition, avec un niveau de détails élevé. Il n'est cependant pas parfait, et contient du bruit et des trous, n'ayant pas pu placer le scanner laser partout. Une deuxième donnée utilisée par cette thèse est un modèle 3D du télescope Bernard Lyot, réalisé à l'aide de Google Sketchup. Ce modèle a ensuite été utilisé pour généré le maillage utilisé par la suite dans cette thèse. Ce maillage présente donc une représentation idéale du télescope, mais avec un niveau de détails et une précision plus faible concernant les mesures du modèle.

De ce projet est née une première thèse, décrite par ce présent manuscrit.

1.3 Organisation du manuscrit

Les environnements 3D sont de plus en plus courants. Ils offrent l'avantage de pouvoir afficher une représentation proche de la réalité. Mais pour construire ces environnements des modèles 3D ont besoin d'être créés, représentant les objets composant le monde. Une première méthode est de les concevoir à la main, à l'aide de logiciels dédiés. Plusieurs types de modèles peuvent ainsi être construits. C'est l'objet du début du chapitre 2, qui explique ce qu'est un environnement 3D et les différentes manières de représenter un objet 3D. Il détaille ensuite les interactions possibles offertes par celui-ci, de la navigation à la sélection.

Mais cette façon de faire prend du temps, et peut rendre difficile d'acquérir tous les détails de la réalité, tant au niveau de la géométrie que des attributs de rendu, tel la couleur. Une méthode alternative existe alors : acquérir numériquement des objets réels. Cette méthode comporte également des problèmes, mais permet d'obtenir des résultats très réalistes. Et cette méthode n'est pas utilisée seulement pour afficher une représentation de l'objet dans un environnement 3D. Elle peut être également utilisée pour sauvegarder un bâtiment, pour en avoir un plan, pour par exemple avoir une base de comparaison après des dégâts matériels subis lors d'un incendie.

Les modèles obtenus à l'aide d'un scanner sont généralement assez com-

plexes. En effet, une acquisition n'est pas forcément une manipulation qui sera réalisée souvent car acquérir un objet nécessite du matériel spécifique, le scanner, mais aussi une préparation pour choisir les endroits où le placer de manière à obtenir tout l'objet et de fusionner l'ensemble des acquisitions, la maîtrise du logiciel qui permet de faire cela... Autant donc obtenir le meilleur modèle possible dès le départ. Cependant le niveau de détail peut être un problème dans certains cas. De même il peut être possible d'avoir scanné plus que nécessaire, au cas où. La première chose à faire est donc de réaliser une sélection dans le modèle obtenu, pour ne garder que la partie qui nous intéresse vraiment. Les différentes manières de réaliser une sélection dans un environnement 3D sont détaillées dans la suite du chapitre 2, en fonction de la représentation du modèle acquis.

Mais la plupart des scanners ne savent acquérir un modèle qu'en nuage de points. En effet, c'est généralement ce que la technologie permet d'obtenir, et la conversion dans une autre représentation n'est pas aisée. C'est pourquoi par la suite nous nous intéressons plus particulièrement aux nuages de points. De plus, les scanners ne sont pas parfaits. En effet, ils possèdent un niveau de précision maximale possible en fonction des capteurs qu'ils utilisent. Ils sont également affectés par d'autres paramètres environnementaux, comme la distance à leur cible ou les conditions atmosphériques. Toutes ces raisons font que le nuage de points acquis n'est pas parfait non plus. Il comporte certaines informations qui sont fausses, ou des trous là où le scanner n'a pas pu acquérir de données. Les différents traitements disponibles dans l'état de l'art pour corriger ces problèmes sont l'objet de la dernière partie du chapitre 2, du remplissage des trous à la simplification pour obtenir un modèle comportant moins de détails.

Cependant le problème que cette thèse essaie de corriger est un peu plus particulier. Son contexte a été expliqué au début de ce chapitre. Il a expliqué pourquoi les méthodes automatiques ne sont pas parfaites, et a poursuivi sur le contexte particulier de cette thèse, initiée à partir d'un projet en collaboration avec l'OMP opérant le TBL situé au sommet du Pic du Midi. Le chapitre 3 présente ensuite la solution proposée, qui consiste à réparer les nuages de points obtenus par scanner 3D à l'aide d'autres données relatives au modèle représenté et déjà existantes, comme par exemple un maillage obtenu à partir d'un modèle fait par conception assistée par ordinateur (CAO). Ces données peuvent ne pas être complètement fiables ; elles sont là pour aider à la correction. Le chapitre 3 présente

ensuite le processus utilisé, comportant quatre étapes. Il termine enfin sur la présentation de la base du le logiciel mettant en œuvre notre processus et en détaille quelques parties intéressantes. Ce processus comporte deux étapes clefs : la sélection et l'amélioration de modèles.

La première étape de notre processus est la sélection. Elle permet de ne traiter que la partie d'un modèle qui nous intéresse, et évite de travailler sur le reste. Des techniques de sélection existent permettant de réaliser cet objectif. Cependant, nous leur avons trouvé des lacunes, et avons donc choisi d'en créer une nouvelle, en espérant trouver un bon compromis entre facilité d'utilisation et flexibilité de la sélection réalisée. Le détail de cette technique de sélection et sa raison d'être sont l'objet du chapitre 4. Il montre également, à l'aide d'une évaluation expérimentale, que cette technique est, entre autres, préférée par les utilisateurs. Son principe est de créer itérativement un volume par un ensemble de sections définies librement, jusqu'à englober la partie à sélectionner.

L'étape de l'amélioration de modèles est elle-même divisée en 3 sous-étapes. Elles sont détaillées dans le chapitre 5. Elles consistent en la détection de zones communes, suivie de la détection de problèmes puis de leur résolution. Elles permettent de découvrir quels problèmes existent dans le nuage de points et comment les corriger. La première chose à faire est de définir quels problèmes il est possible de rencontrer. Une fois cela fait, il faut pouvoir les détecter ; c'est le but des deux premières étapes, qui trouvent une liste de problèmes présent dans le modèle à améliorer. Ces problèmes sont alors corrigés dans la dernière étape. De plus, ce processus se veut général ; bien que dans cette thèse uniquement l'amélioration de nuages de points soit abordée, en théorie le processus est capable de corriger les problèmes de n'importe quelle représentation, après adaptation de celui-ci. Ce chapitre montre que le processus donne de bons résultats sur les nuages de points testés.

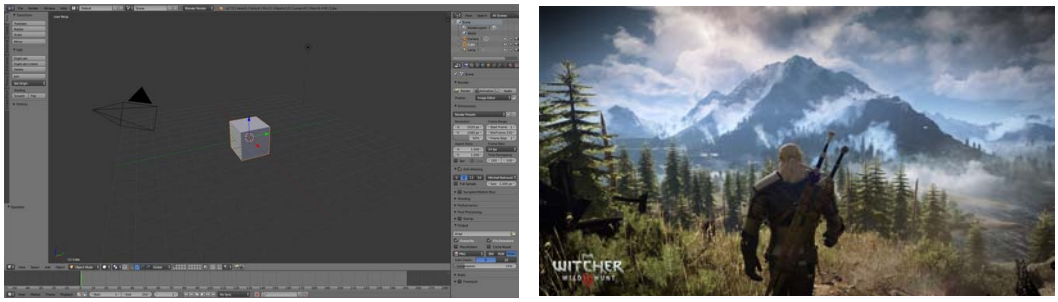
Enfin, un bilan final des contributions et de leurs perspectives est dressé dans la partie conclusion du présent mémoire.

Chapitre 2

État de l'art

Le domaine de l'informatique 3D est vaste et comporte de nombreuses recherches déjà accomplies. Il est également défini par des mots clef utilisés couramment. Ce chapitre vise à faire une introduction du domaine et donner les éléments nécessaires à la compréhension de cette thèse. C'est pourquoi le contexte commence par être mis en place avec la définition d'un environnement 3D et des modèles le composant, pour se focaliser plus sur les concepts qui sont discutés par la suite, de la sélection dans un tel environnement aux traitements réalisés sur les modèles.

2.1 Environnement 3D



(a) Blender, un modeleur 3D [Blender Foundation 1995]

(b) The Witcher 3, un jeu vidéo [Projekt 2015]

Figure 2.1 – Exemples d'environnements virtuels 3D

La simulation d'environnements virtuels par un ordinateur est utile pour de nombreuses tâches. Ces environnements ne sont pas limités par les contraintes physiques du monde réel, et peuvent prendre de nombreuses formes. Un environnement virtuel 3D est alors une représentation d'un ensemble d'objets virtuels, avec lesquels des interactions sont possibles. Les interactions permises

dépendent de l'application. Par exemple, dans le cas d'un modeleur 3D comme Blender (figure 2.1a), il sera possible de faire un grand nombre d'actions, allant du contrôle du point de vue à la modification des modèles représentés. Un autre exemple d'environnement virtuel 3D est celui d'un jeu vidéo (figure 2.1b), qui typiquement autorisera uniquement des interactions via un avatar dans le jeu. Ainsi, le contrôle du point du vue sera indirectement réalisé à partir du contrôle de l'avatar.

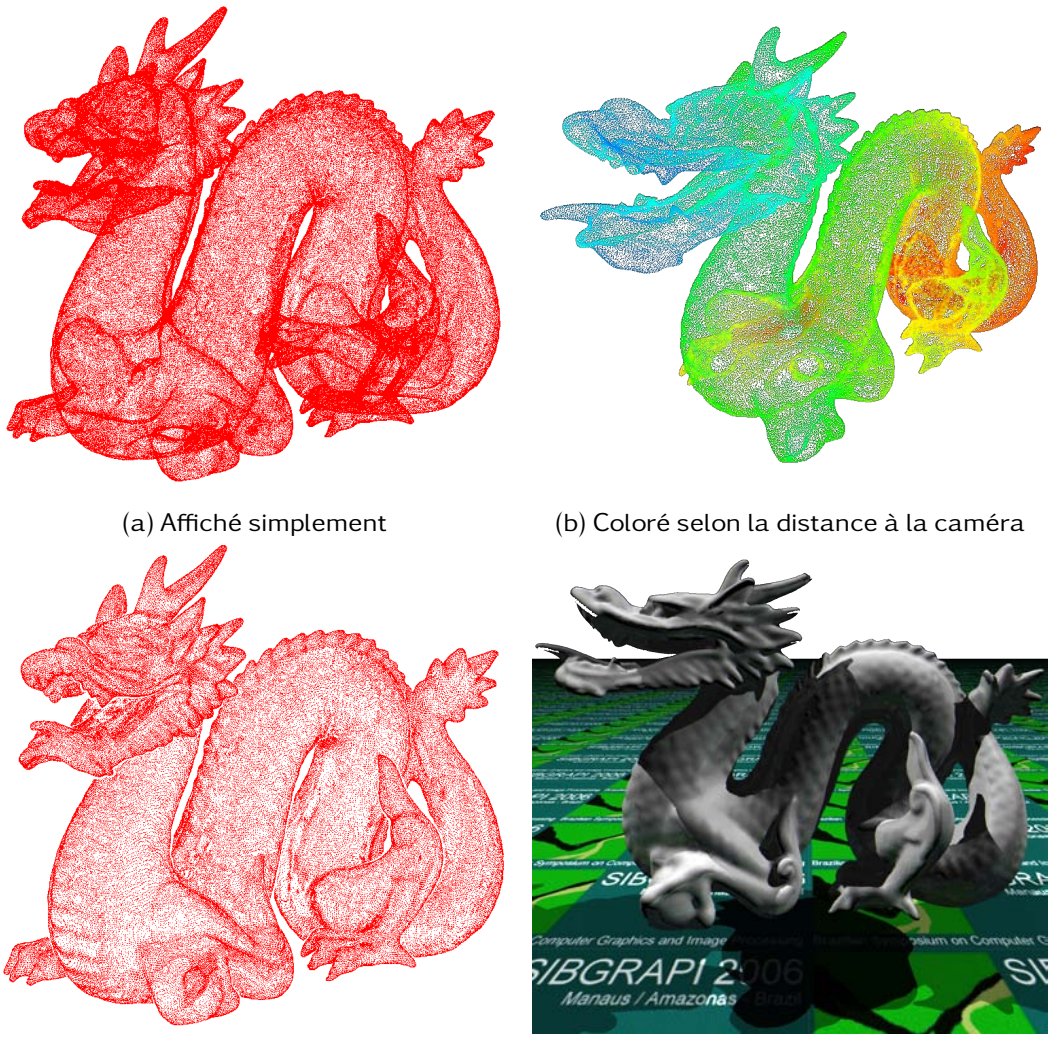
2.1.1 Modèles 3D

Les représentations des modèles 3D peuvent prendre de nombreuses formes. Dans cette section sont détaillées les plus courantes. Toutes les représentations possèdent des forces et des faiblesses, ce qui explique que le choix d'une représentation dépende fortement du contexte dans lequel elle va être utilisée. Cependant la plus couramment utilisée reste le maillage, car elle offre un compromis satisfaisant entre précision, espace de stockage et rapidité d'affichage. De plus la grande majorité des cartes graphiques disponibles dans le commerce sont spécialisées pour traiter les maillages, bien que toutes les représentations détaillées ici soient affichables par celles-ci.

2.1.1.1 Nuage de points

Comme son nom l'indique, un nuage de points est un ensemble de points. Chaque point indique un endroit où la surface de l'objet représenté existe. L'information est donc stockée par la position relative de chaque point les uns par rapport aux autres. Elle peut être, dans certains cas, associée à une couleur pour chaque point ou une normale, représentant la direction de la surface au niveau du point. La figure 2.2 montre un exemple de nuage de points représentant un dragon. Le modèle est tiré du *Stanford Computer Graphics Laboratory*. Les couleurs de la figure 2.2b représentent la distance du point affiché à la caméra, allant du bleu (le plus proche) en passant par le vert jusqu'au rouge (le plus éloigné).

Propriétés Un nuage de points ne contenant que des points, cette représentation a l'avantage d'être très simple à stocker et afficher. Le problème principal vient que la surface représentée est inconnue ; en effet, comment savoir si deux



(a) Affiché simplement

(b) Coloré selon la distance à la caméra

(c) Cachant les points arrières

(d) Utilisant un rendu avancé

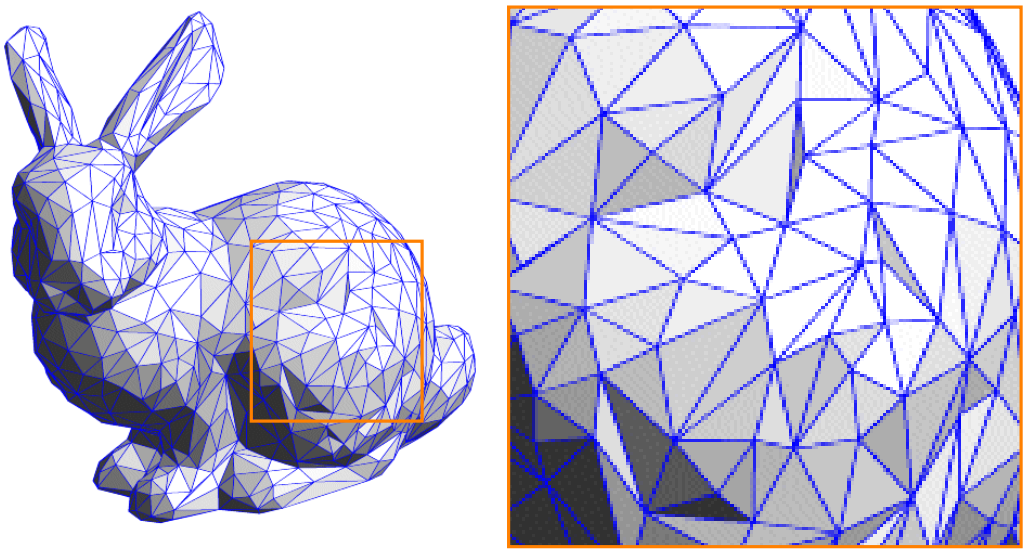
FIGURE 2.2 – Nuage de points : Dragon du *Stanford Computer Graphics Laboratory*

points appartiennent à la même surface ? Cela rend difficile le calcul des normales par exemple, qui sont utiles pour la segmentation, entre autres. Un autre problème concerne le rendu du nuage de points, car si le nombre de points est faible, des trous seront visibles. La figure 2.2a montre bien que la surface est parfaitement représentée lorsqu'il y a assez de points, mais que des trous sont visibles. Cela peut être compensé par plusieurs algorithmes de rendu qui permettent d'afficher un nuage de points de manière à observer l'objet représenté de manière très réaliste. Par exemple [Gois *et al.* 2006] utilisent un opérateur de projection basé sur la courbure, la normale et une équation de diffusion anisotrope afin de créer une surface lors de la visualisation, créant une image comme celle visible sur la

figure 2.2d. Les différences avec un maillage ne sont pas visibles. Cependant, si le but est d'afficher les points, ceux étant à l'arrière du modèles deviennent visibles (figure 2.2a). Il devient donc difficile de comprendre ce qui est affiché, car il n'est pas possible de discerner quels sont les points devant ou derrière. Une solution est de les colorer selon leur distance, comme sur la figure 2.2b, ce qui permet d'améliorer légèrement la compréhension. Une autre solution est de cacher les points qui ne seraient pas visibles si la surface existait. Pour cela, [Katz *et al.* 2007] utilisent un opérateur se basant sur l'enveloppe convexe du nuage de points ainsi que sur le point de vue courant, obtenant le résultat visible sur la figure 2.2c.

Acquisition Un nuage de points n'est en général pas créé avec un logiciel de CAO par un humain. En effet, cette représentation est celle qui est fournie par la plupart des scanners qui permettent d'obtenir un modèle 3D à partir d'un objet réel. Typiquement un nuage de points créé ainsi contiendra un grand nombre de points, pouvant facilement atteindre et dépasser la centaine de millions selon l'objet représenté, de façon à obtenir un grand niveau de détails. C'est aussi un nuage de points qui sera obtenu lorsque une scène 3D sera reconstruite à partir de plusieurs photos. Les nuages de points ont donc tendance à représenter un objet réel et à contenir des défauts, dus à la manière dont ils sont construits. Les défauts sont en général des trous (des zones où il manque des points) ou du bruit (des points ne représentant pas la surface).

Il existe deux catégories de nuages de points selon la manière avec laquelle ils ont été acquis. La première est les nuages de points non-ordonnés. Ils contiennent des points mais dans aucun ordre particulier, ni ne formant une quelconque structure. C'est un peu comme si les points avaient été placés aléatoirement sur la surface. La deuxième catégorie est alors les nuages de points ordonnés. Ils sont créés par les scanners, qui acquièrent les modèles en balayant un objet de manière régulière, créant des nuages de points où tous les points sont alignés sur une grille 2D. Certains algorithmes sont capables de prendre cet aspect en compte pour les aider à accomplir leur tâche [Sithole & Vosselman 2003].



(a) Modèle (b) Agrandissement

FIGURE 2.3 – Maillage : Lapin du *Stanford Computer Graphics Laboratory*

2.1.1.2 Maillage

Alors qu'un nuage de points ne contient que des points, un maillage reprend cette base mais en y ajoutant des liens. Ainsi, un maillage est en fait constitué de polygones. La grande majorité des maillages est constituée de triangles, qui constituent les polygones les plus simples, mais peuvent être en théorie composés de n'importe quel type de polygone. Le principe est alors de placer de nombreux polygones côte à côte, partageant leurs arêtes pour éviter les trous, représentants une approximation de la surface de l'objet représenté. Un exemple de maillage est visible figure 2.3a, avec un agrandissement qui montre les triangles qui le composent figure 2.3b.

Propriétés L'avantage de cette représentation est de rester globalement simple à stocker et afficher mais contenant plus d'informations qu'un nuage de points, étant donné que la surface du modèle est connue. Cette représentation reste cependant une approximation de la surface, peu d'objets pouvant être parfaitement représentés avec un ensemble de polygones. C'est aussi pour traiter cette représentation que les cartes graphiques ont été conçues, elles sont donc capables d'afficher de nombreux triangles en temps réel pour afficher une scène visuellement réaliste.

Création De nombreux logiciels existent permettant de créer des maillages. Le principe est de construire la surface de l'objet avec des triangles. Bien que le principe de créer les triangles un par un peut sembler fastidieux, en pratique ces logiciels offrent de nombreuses fonctions avancées pour aider à la construction de modèle. Par exemple, il est possible de ne créer qu'une moitié d'un modèle représentant un humain et de générer automatiquement l'autre moitié par symétrie. Une autre approche utilisée pour la création de maillage est d'utiliser le principe de la sculpture. La création consistera alors à commencer par faire une forme très grossière du modèle, pour ensuite venir appliquer différents outils pour creuser le modèle ou rajouter de la « matière » [Pixologic].

2.1.1.3 Objet de forme libre (CAO)

Les objets de forme libre sont composés d'un ensemble de formules mathématiques. Il existe plusieurs manières de les représenter. Deux façons sont courantes. La première est ce qu'on appelle les modèles CSG. Elle consiste à prendre des formes simples définies mathématiquement comme des sphères ou des cubes, puis à créer l'objet à représenter par une succession d'opérations booléennes (union, différence, intersection) ou par des transformations (translation, rotation, déformation) de ces formes. Par exemple, il est possible de créer un trou dans un cube si on fait une différence avec une sphère. Cette représentation définit donc un volume. La deuxième méthode est la représentation par les modèles B-Rep. Cela consiste à représenter la frontière d'un modèle, c'est-à-dire la limite entre l'intérieur et l'extérieur. Plusieurs types de surfaces sont ainsi utilisées pour cela, comme les B-Splines, les surfaces de Bézier ou les NURBS, qui sont des formules mathématiques utilisant ce qui est appelé « points de contrôle ». Les points de contrôle permettent de définir la surface simplement. Un modèle B-Rep est alors un modèle constitué de ces surfaces, utilisées pour former la frontière du modèle.

Propriétés L'avantage des objets de forme libre est qu'ils peuvent représenter l'objet voulu exactement, sans approximation. Cependant l'utilisation des formules mathématiques peuvent rendre un tel objet difficile à manipuler et à afficher efficacement. La solution généralement retenue pour les afficher est de d'abord les convertir en un maillage. Un autre avantage de cette représentation est qu'elle peut être utilisée pour faire des simulations précises sur les objets re-

présentés, permettant par exemple de savoir à quel niveau de pression un objet va casser.

Création De la même manière que pour les maillages, de nombreux logiciels existent pour créer des objets de forme libre. Ces logiciels sont généralement utilisés par des professionnels pour des projets d'ingénierie [Systèmes a,Systèmes b].

2.1.1.4 Autres représentations

De nombreuses autres représentations existent. Par exemple, dans le domaine médical les images 3D, composées de voxels, sont beaucoup utilisées. Un voxel est l'équivalent d'un pixel pour une image 3D. Une image 3D est alors une grille régulière cubique dont chaque case possède une valeur. Cette valeur peut représenter n'importe quelle grandeur, mais dans le domaine médical cette grandeur est généralement une densité. Les images 3D ont l'avantage d'être très simples à manipuler mais sont très coûteuses en mémoire et un peu plus complexes à afficher qu'un maillage (ce qui dépend de la grandeur contenue). Les grandeurs sont en général volumiques, car même s'il est possible de stocker une surface à l'intérieur cela laisserait beaucoup de cases vides.

Les cartes de profondeurs sont aussi une autre représentation, même s'il serait plus approprié de parler de 2.5D. En effet, les cartes de profondeur sont des images classiques mais qui enregistrent une distance par pixel. La 3D n'est donc valable qu'à partir du point de vue de l'image, et est autrement très incomplète. C'est un format renvoyé par la caméra de Kinect par exemple.

2.1.1.5 Conversion

Un même objet peut être modélisé à l'aide de plusieurs représentations, mais le choix n'est pas forcément définitif. En effet, il est possible de changer la représentation d'un objet ; c'est ce qui est appelé la conversion. En revanche ce processus n'est pas forcément aisé. En effet, s'il est relativement simple de passer d'un maillage à un nuage de points par exemple, l'opération inverse est difficile ; il faut retrouver la surface automatiquement. De la même manière pour convertir en objet de forme libre il faut être capable de retrouver les formules mathéma-

tiques représentant l'objet, aussi appelé rétro-ingénierie, ce qui est difficile. Ce processus peut entraîner une perte d'information.

2.2 Interactions dans un environnement 3D

Selon [Bowman *et al.* 2001], il existe trois types d'interaction possibles dans un environnement 3D : la navigation, la sélection et manipulation et enfin le contrôle du système. Plus récemment, [Jankowski & Hachet 2013] ont réalisé un état de l'art de ce domaine. Cette section fait un résumé de leur travaux et des trois types d'interaction possibles.

2.2.1 Navigation

Du fait de la taille d'un environnement 3D, il n'est en général pas possible de le visualiser entièrement à partir d'un seul point de vue. Il faut donc la plupart du temps mettre en œuvre la navigation qui permettra de remplir différents objectifs en déplaçant le point de vue sur l'environnement 3D pour pouvoir en visualiser tous ses aspects. Cependant se déplacer n'est pas une tâche aisée. Une caméra quelconque aura 6 degrés de liberté à manipuler, c'est-à-dire les translations et les rotations des trois axes de l'environnement. Il y a de plus des paramètres à régler, car une personne faisant un travail de précision sur un modèle 3D ne voudra pas la même vitesse de déplacement qu'une personne simplement en train de visiter un environnement. Il y a aussi le problème de trouver son chemin. En effet, de la même manière qu'il est possible de se perdre dans une ville, il est possible de se perdre dans un environnement 3D. Selon [Mackinlay *et al.* 1990], quatre types de mouvements sont alors définis pour un besoin de navigation :

- un mouvement d'exploration, qui consiste à simplement marcher dans l'environnement, par exemple pour établir ses repères. C'est ce type de mouvements que nous pouvons retrouver dans le jeu *Gone Home* [Company 2013] ;
- un mouvement ciblé, qui aura pour but de trouver une cible spécifique. C'est le type de tâche de la première expérience de [Bergé *et al.* 2014] ;
- un mouvement à des coordonnées spécifiques, par exemple pour enregistrer un point de vue. C'est une fonction que nous pouvons retrouver dans les logiciels comme Blender, qui permettent d'aligner la vue avec un axe de

l'environnement 3D ;

— un mouvement suivant une trajectoire (incluant l'orientation) spécifique, comme les mouvements d'une caméra de cinéma. C'est donc quelque chose qui sera très utilisé pour les films d'animation comme *Toy Story* [Pixar 1995].

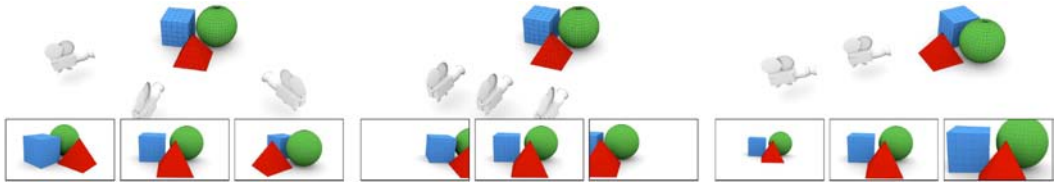


FIGURE 2.4 – Mouvement de navigation de [Jankowski & Hachet 2013], rotation, translation, zoom

Ces quatre types de mouvement ont chacun fait l'objet de travaux, qui ont aidé à déterminer des manières de procéder pour les réaliser. Par exemple, pour le mouvement d'exploration, la distinction est faite entre les interactions de type rotation-translation dans le plan de la caméra-zoom et les interactions de type simulation de marche, de conduite ou de vol (qu'il est possible de retrouver dans les jeux vidéos). La figure 2.4 montre différentes manières de déplacer la caméra. De gauche à droite, elle montre les mouvements de rotation, translation et zoom, avec situés en haut l'environnement 3D et les positions successives de la caméra, et en bas des aperçus de rendu avec celle-ci.

2.2.2 Sélection et manipulation

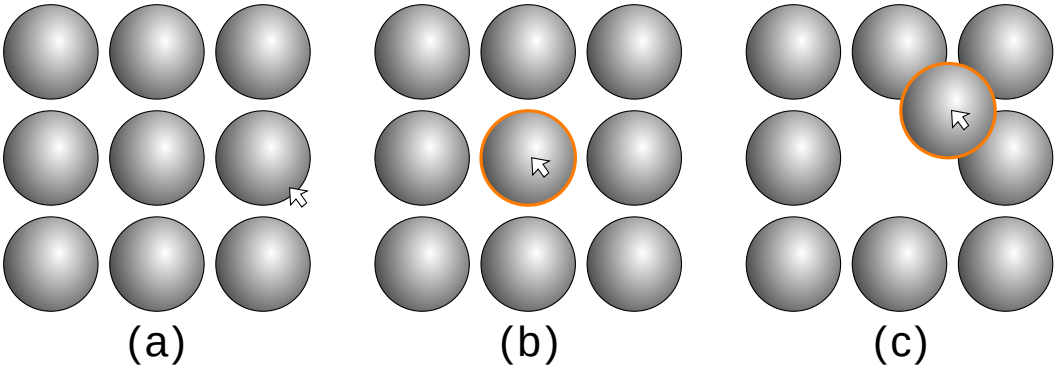


FIGURE 2.5 – Exemple de sélection et manipulation

La sélection intervient naturellement après la navigation. Une fois que le modèle 3D d'intérêt est visible, il faut pouvoir interagir avec. Mais pour pouvoir interagir, il faut d'abord déterminer la cible : c'est la sélection. Dans un environne-

ment 3D, la sélection se fait généralement avec la souris. Le curseur pointe sur un objet, et un clic permet de désigner le modèle pointé comme cible de la sélection [Elmqvist & Fekete 2008]. Cela peut être plus compliqué, en fonction de la représentation ou de la cible. Typiquement sélectionner par un clic ne fonctionne pas avec un nuage de points (voir section 2.3). La section suivante détaille plus le contexte particulier de la sélection dans un nuage de points.

Une fois le modèle sélectionné, il reste à définir l'interaction. La manipulation peut simplement consister à déplacer ou changer l'orientation du modèle, mais elle peut aussi venir le déformer en fonction des objectifs [Nielson & Olsen 1987]. Un exemple de sélection et manipulation d'un objet est visible sur la figure 2.5. Sur la première image 2.5a il existe neuf sphères. La deuxième image 2.5b montre le curseur s'étant déplacé et ayant sélectionné la sphère du milieu, comme indiqué par son contour orange. Enfin, la troisième image 2.5c montre cette sphère déplacée avec le curseur.

2.2.3 Contrôle du système

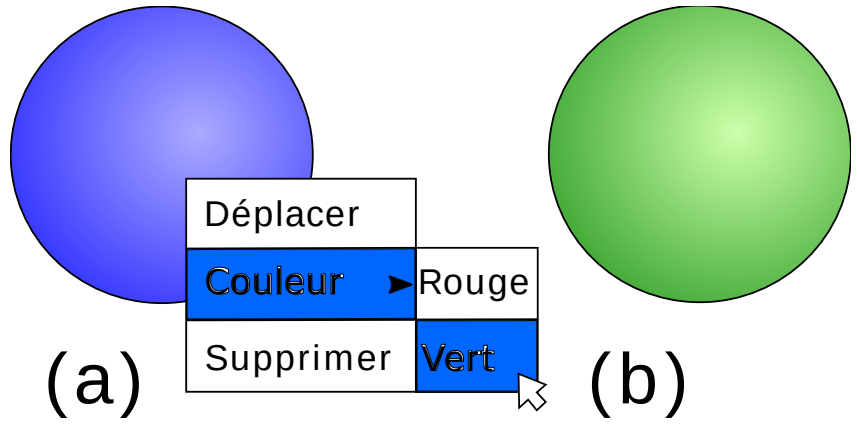


Figure 2.6 – Exemple de contrôle du système

Enfin, la dernière interaction possible avec un environnement 3D est le contrôle de l'application [Bowman *et al.* 2001]. Cela signifie que c'est une interaction qu'il n'est pas possible de faire à l'intérieur de l'environnement mais qui va en changer les propriétés. Typiquement, dans un modeleur 3D, c'est ce type d'interaction qui va permettre de passer de l'affichage d'un modèle polygonal à l'affichage des polygones le composant. Un autre exemple est visible figure 2.6. Dans la première image, 2.6a, une sphère de couleur bleue est visible. Un menu est alors ouvert,

qui permet de changer la couleur en vert, comme sur la deuxième image 2.6b.

2.3 Sélection dans un nuage de points

De nombreuses approches existent pour effectuer une sélection dans un environnement 3D. Le but de celle-ci est de pouvoir désigner une partie de la scène 3D, afin de pouvoir la manipuler sans toucher au reste. La manipulation peut prendre la forme d'un déplacement, d'une rotation, d'une suppression... Cela est libre et dépend de l'application. Dans le cas traité par notre processus, la sélection permet de désigner sur quelle partie le travail sera effectué, ignorant le reste. Le but est donc d'éviter de travailler sur des parties qui ne présentent pas d'intérêt particulier et de gagner ainsi du temps de calcul. Nous nous intéressons uniquement à la sélection dans un nuage de points.

Cette sélection est un peu particulière pour deux raisons. La première est que le but n'est pas de sélectionner un modèle 3D, mais une partie de celui-ci. Il faut donc être capable de désigner chaque point à sélectionner individuellement. La deuxième raison est le fait qu'il n'y ait que des points et pas de surface, ce qui rend la sélection difficile pour un utilisateur. Cela signifie que, de la même manière que deux étoiles dans le ciel apparaissent proches, deux points dans un nuage de points peuvent en réalité être très éloignés l'un de l'autre quand ils sont affichés côte à côte à l'écran. Pour atténuer ce problème, des techniques de rendus peuvent être utiles [Katz *et al.* 2007] mais ne sont pas l'objet de ces présents travaux. Par défaut, il est inévitable de combiner simultanément le contrôle de la technique de sélection et du point de vue de la caméra. Les techniques interactives existantes synthétisées dans cette section possèdent toutes cette dualité.

2.3.1 Segmentation automatique

Utiliser un algorithme de segmentation constitue une première solution pour sélectionner automatiquement les points d'un nuage. Cela permet de s'affranchir du problème de la manipulation de la caméra. Plusieurs classes d'algorithme de segmentation existent [Wang & Shan 2009] :

— la détection par arête, qui convertit les nuages de points en images de profondeur et applique une technique de segmentation d'image pour trouver

les arêtes, mais la conversion perd de l'information ;

— la croissance de région, qui consiste à placer des graines et étendre des région à partir de celles-ci selon un critère donné, comme la similitude de l'orientation de la surface par exemple. Cette méthode a l'avantage d'être simple et relativement légère en calculs, mais dépend fortement de l'emplacement initial des graines qui n'est pas trivial à déterminer ;

— le suivi des lignes de scanner, qui profite du motif en ligne de certains nuages de points, mais qui n'est pas directement applicable aux nuages de points non-ordonnés,

— le regroupement de points, qui convertit chaque point en un vecteur caractéristique déterminé par exemple en fonction de propriétés géométriques, et réalise la segmentation sur ces vecteurs. Les performances dépendent alors beaucoup de la construction du vecteur caractéristique ;

— le partitionnement de graphe, qui converti le nuage de points en un graphe et utilise un algorithme de partition de graphe.

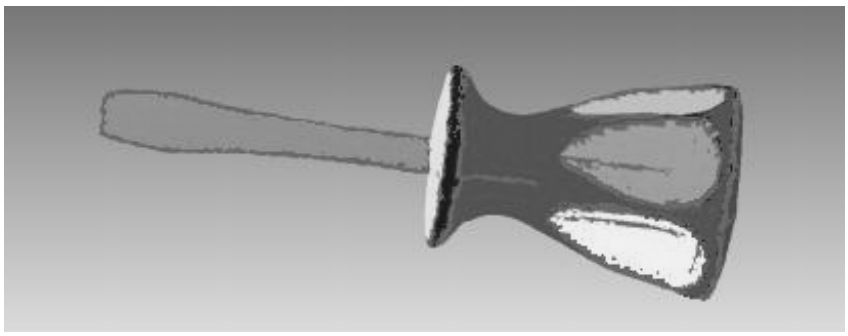


FIGURE 2.7 – Résultat de segmentation de [Stamati & Fudos 2007]

Les trois classes les plus courantes sont la croissance de région, la partition de graphe et le regroupement de points. [Stamati & Fudos 2007] par exemple utilisent une croissance de région. Ils utilisent comme critère pour étendre leurs régions l'angle entre la normale du point courant et celle de la région (calculée en prenant la moyenne de toutes ses normales) et le gradient de leur fonction, indice sur le fait que le modèle est concave ou non. Un exemple de résultat obtenu par leur méthode est visible figure 2.7.

[Ma *et al.* 2010] quant à eux ont choisi d'utiliser un algorithme de partition de graphe. Ils construisent le graphe en connectant les points voisins entre eux, et assignent un poids aux arêtes correspondant à un critère de similarité. Ils utilisent ensuite un algorithme de *min-cut* pour obtenir leur segmentation. Un exemple de

FIGURE 2.8 – Résultat de segmentation de [Ma *et al.* 2010]

résultat obtenu par leur méthode est visible figure 2.8.

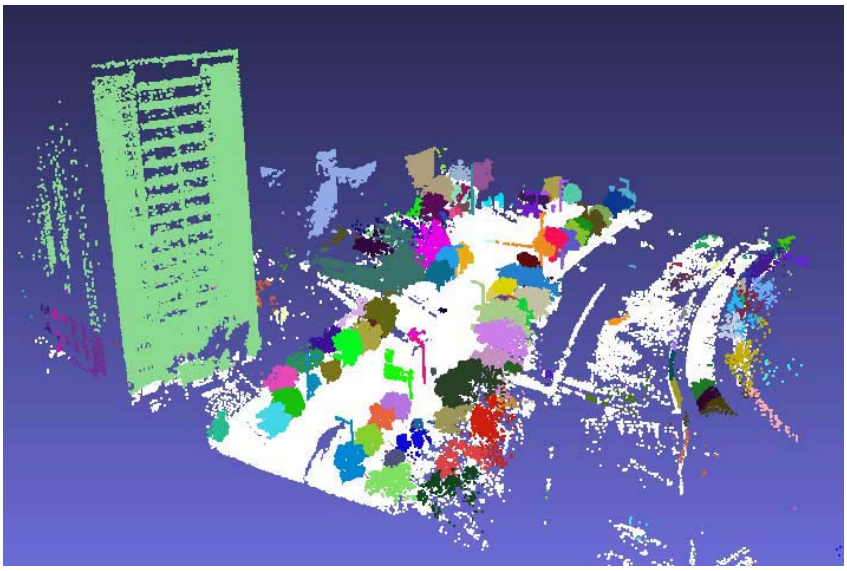


FIGURE 2.9 – Résultat de segmentation de [Yu *et al.* 2011]

Enfin, la classe utilisée par [Yu *et al.* 2011] est le regroupement de points. Après avoir détecté et retiré le sol de leur nuage de points à traiter, les auteurs utilisent une méthode proche du *mean-shift* mais plus spécialisée pour mieux fonctionner dans leur cas d'étude (les nuages de points de rues capturés par des voitures se déplaçant). Un exemple de résultat obtenu par leur méthode est visible figure 2.9.

Une autre approche de la segmentation automatique consiste à utiliser des informations déjà connues sur l'objet recherché. [Nan *et al.* 2012], par exemple,

FIGURE 2.10 – Résultat de segmentation de [Nan *et al.* 2012]

commencent par entraîner un algorithme de classification de modèle. Les auteurs obtiennent ainsi une référence de modèle générique, qu'ils vont ensuite rechercher dans leur scène 3D. De plus, leur algorithme est capable de détecter les objets ressemblant en adaptant leur référence au nuage de points. Un exemple de résultat obtenu par leur méthode est visible figure 2.10. Le problème de cette méthode est qu'il faut connaître en avance l'objet recherché.

Cependant utiliser un algorithme de segmentation automatique pour réaliser une sélection n'est pas chose aisée. En effet, certains algorithmes peuvent nécessiter l'utilisation de pré- ou de post- traitements permettant d'enlever le bruit ou de fusionner des zones, afin d'obtenir de meilleurs résultats. De plus, même une fois ces premiers problèmes réglés, il est peu probable d'obtenir exactement le résultat voulu. Cela peut se voir sur les figures 2.7 ou 2.8 par exemple, qui ne donnent pas forcément des résultats intuitifs pour une personne. Des points non cibles sont inclus, ou des points cibles sont manqués, les paramètres ont besoin d'être ajustés... Il n'est également pas évident de trouver l'algorithme approprié pour son besoin, chacun ayant ses forces et faiblesses selon le type de modèle ou le type de sélection.

Une autre solution est alors d'utiliser une approche interactive, offrant la possibilité à un humain de réaliser la sélection.

2.3.2 Sélection basée rayon

Le lancer de rayon est une technique classique dans la littérature pour effectuer une sélection sur un maillage [Steinicke *et al.* 2004]. [Lee *et al.* 2003] montrent que la technique a l'avantage d'être simple et intuitive, comme la souris. Elle permet aussi de garder une complexité faible, car seulement un clic de la souris est nécessaire pour envoyer un rayon. Elle consiste à déterminer un rayon partant du point de vue de l'utilisateur (la position de la caméra dans la scène 3D) et passant par le point où le clic de la souris a eu lieu. Le rayon traverse ainsi la scène 3D, et l'objet le plus proche du point de vue utilisateur intersectant le rayon devient ainsi sélectionné.

Bien que cette technique fonctionne bien, elle n'est pas adaptée à la sélection d'un nuage de points. En effet, un nuage de points ne possédant pas de surface, aucune intersection n'aura lieu avec le rayon et donc aucune sélection. Elle n'est de plus pas adaptée à la sélection d'une sous-partie, même pour un maillage. Cette sélection nécessiterait de lancer un rayon par triangle, rendant la tâche fastidieuse.

FIGURE 2.11 – Sélection de [Veit & Capobianco 2014]

[Veit & Capobianco 2014] ont quand même essayé d'adapter le principe, en pré-définissant des boîtes basées sur un *octree* englobant le nuage de points.

Ils demandent ensuite à l'utilisateur de pointer vers une boîte (représentant une feuille de l'*octree*), comme nous le montre la figure 2.11. D'autres, comme [Cashion & LaViola 2014] ont décidé de remplacer le rayon par un cône. Cela permet de garder les avantages du lancer de rayon tout en rendant son utilisation possible pour un nuage de points, en sélectionnant tout ce qui se trouve à l'intérieur. Cependant utiliser un cône réduit aussi la précision, le curseur ayant grandi.

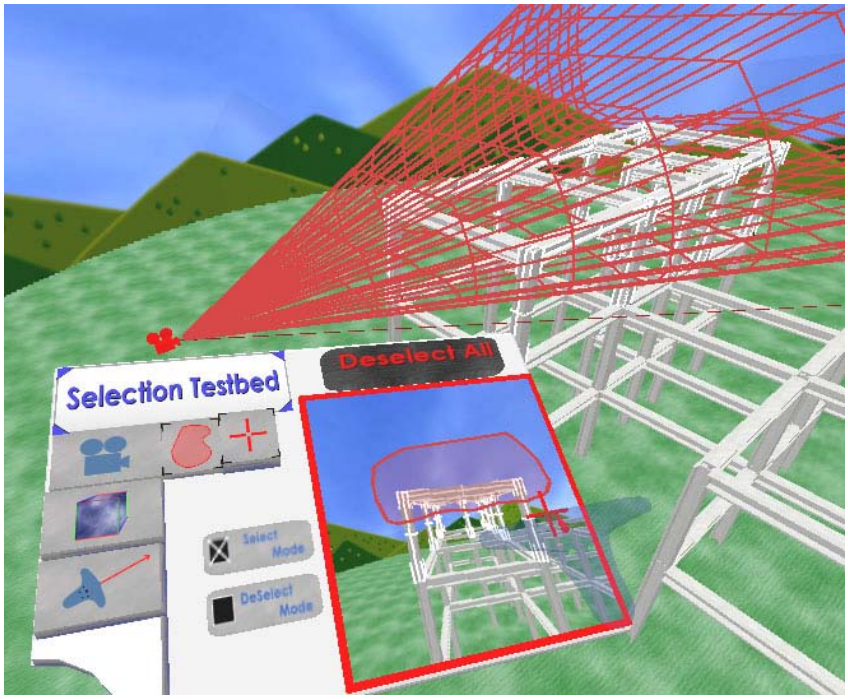


FIGURE 2.12 – Sélection par dessin de [Lucas *et al.* 2005]

Le volume utilisé n'est pas forcé de rester fixe. [Cashion & LaViola 2014] offrent une plus grande flexibilité en faisant varier la taille de leur cône en fonction de la vitesse de déplacement du curseur, en plus de modifier son orientation. [Lucas *et al.* 2005] vont plus loin et proposent de dessiner n'importe quelle forme fermée à l'écran ; comme le montre la figure 2.12. La forme dessinée est ensuite projetée en profondeur, sélectionnant tout ce qui se trouve à l'intérieur. Dessiner un cercle avec cette technique revient donc à faire la sélection avec un cône. [Yu *et al.* 2012] proposent une approche similaire mais affinent leur sélection en ne prenant en compte que les zones au dessus d'une densité paramétrable. Cette technique fonctionne bien dans leur cas, où leur nuage de points représente un ensemble d'étoiles, mais ne fonctionne pas dans le cas d'un nuage de points provenant d'un scanner car la densité dépend de la distance de l'objet au scanner et

ne représente donc pas une information fiable pour ajuster la sélection.

2.3.3 Sélection basée volume

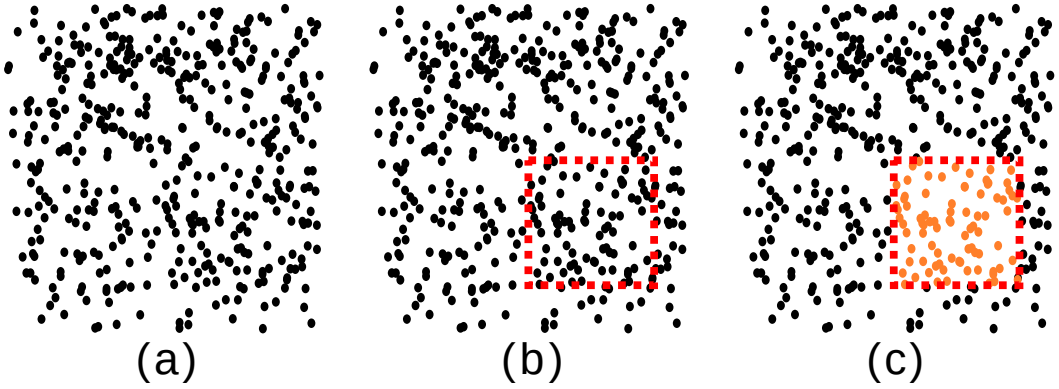


FIGURE 2.13 – Exemple du principe de sélection en 2D

Une autre approche est d'utiliser un volume pour réaliser la sélection, sans projection. Cette approche consiste à placer un volume dans l'environnement 3D. Ce volume peut être de n'importe quel nature : une sphère, un cube, une patate... Le principe est que chaque élément se trouvant à l'intérieur de ce volume devient sélectionné. La figure 2.13 montre le principe en 2D. Il y a tout d'abord un ensemble de points, visible en 2.13a. Un carré est ensuite introduit en 2.13b. Enfin, tous les points à l'intérieur deviennent sélectionnés (figure 2.13c).

Le problème devient alors de contrôler la caméra dans la scène 3D en plus d'offrir la possibilité d'effectuer une sélection. Ce qui peut amener à une solution très simple : l'utilisation d'un volume fixe. Le volume n'est pas paramétrable, et reste toujours à la même position relative à la caméra [Cabral *et al.* 2014]. Mais en plus de rendre la sélection difficile à visualiser et comprendre, la technique requiert de nombreux ajustements avant d'obtenir la sélection désirée.

Une solution pour augmenter la précision de la sélection est alors d'offrir des volumes paramétrables. Une sphère n'offre que deux paramètres possibles, sa position (elle-même possédant trois composantes) et sa taille, elle reste donc simple à manipuler. [Benko & Feiner 2007] en utilisent une pour sa technique de sélection, avec une manière innovante de la contrôler. La scène 3D est affichée en réalité augmentée au dessus d'une table qui représente le sol. La sphère de sélection devient alors un ballon attaché à une ficelle qui essaie de s'envoler, et les doigts

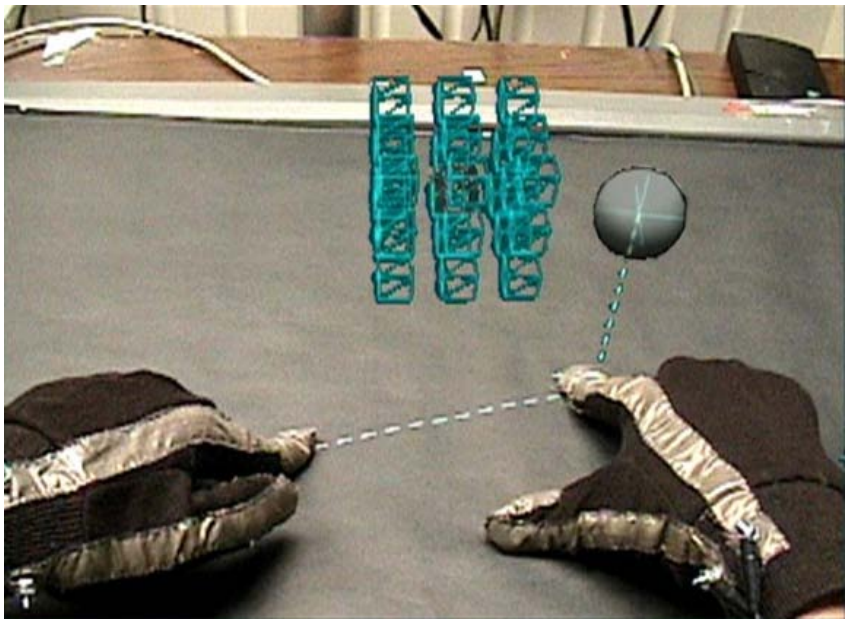


FIGURE 2.14 – Sélection de [Benko & Feiner 2007]

sont utilisés pour le contrôler en les plaçant sur la table. Dans leur technique, la main droite permet de tenir le ballon (donc de définir sa position dans un plan parallèle à la table) et d'en définir la taille (par la distance entre le pouce et l'index). La main gauche tire plus ou moins sur la « ficelle » pour en définir l'altitude. Tous les paramètres de la sphère sont ainsi contrôlés, comme le montre la figure 2.14.

[Naito *et al.* 2009] ont aussi choisi d'utiliser une sphère pour effectuer une sélection, ainsi que la réalité augmentée pour afficher leur scène 3D. Ils utilisent un cylindre transparent pour définir les limites de leur scène, l'affichage étant réalisé à l'intérieur, et se servent de la surface de ce cylindre en tant que dispositif d'interaction (figure 2.15). Pour eux, la sphère représente l'équivalent du pointeur de la souris en 3D, un curseur. La position du curseur est définie par la position des doigts sur le cylindre. Les deux mains posées forment une ligne où la sphère se trouve. Ensuite, par des mouvements de doigts il est possible de contrôler la taille de la sphère mais aussi de l'attirer vers l'une ou l'autre main, déterminant sa position sur la ligne imaginaire.

Une autre forme couramment utilisée dans la littérature est le cuboïde (parallélépipède rectangle), qui offre plus de paramètres contrôlables. [Lucas *et al.* 2005] proposent aussi, en plus de leur technique par dessin décrite à la section précédente, d'utiliser cette forme pour réaliser une sélection. Leur technique consiste

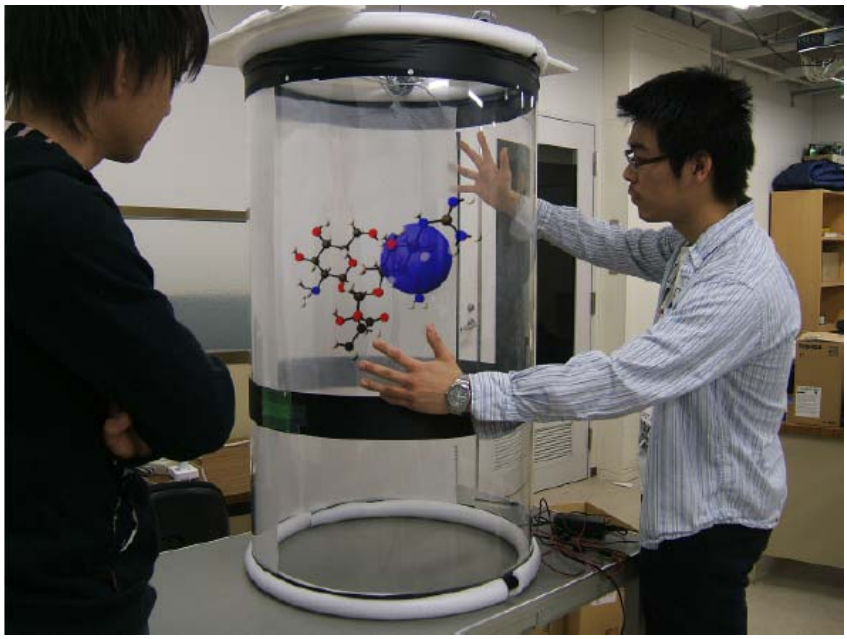


FIGURE 2.15 – Sélection de [Naito *et al.* 2009]

à choisir l'un des trois axes des plans parallèles du cuboïde, puis de pousser ou tirer pour déplacer les faces dans la direction choisie. Pour cela ils utilisent une baguette, dont la position et l'orientation dans l'espace sont capturées, comportant quatre boutons et un joystick.

[Ulinski *et al.* 2007] utilisent également un cuboïde, et ont testé 3 techniques différentes utilisant la position et l'orientation des deux mains dans l'espace pour effectuer une sélection sur un volume (figure 2.16). La première utilise la main non-dominante pour contrôler la position et l'orientation du cuboïde, et la main dominante contrôle la taille dans les trois dimensions. La deuxième technique est similaire mais utilise le centre du cuboïde comme base pour la main non dominante, alors que la précédente utilise un coin. Enfin, la troisième technique fait directement correspondre deux coins en diagonales du cuboïde dans les deux mains, celles-ci changeant donc la forme de la sélection en se déplaçant.

Une troisième forme utilisée dans la littérature est l'ellipsoïde. C'est la forme choisie par [Krammes *et al.* 2014]. Leur technique consiste en un système immersif, avec un visiocasque (écran au niveau des yeux, qui fait tourner la scène 3D en fonction de l'orientation de la tête), un smartphone contrôlant l'interaction et un ensemble de balances pour détecter les pas. L'utilisateur se déplace ainsi en regardant dans la direction voulue et en marchant sur place (figure 2.17a). Il

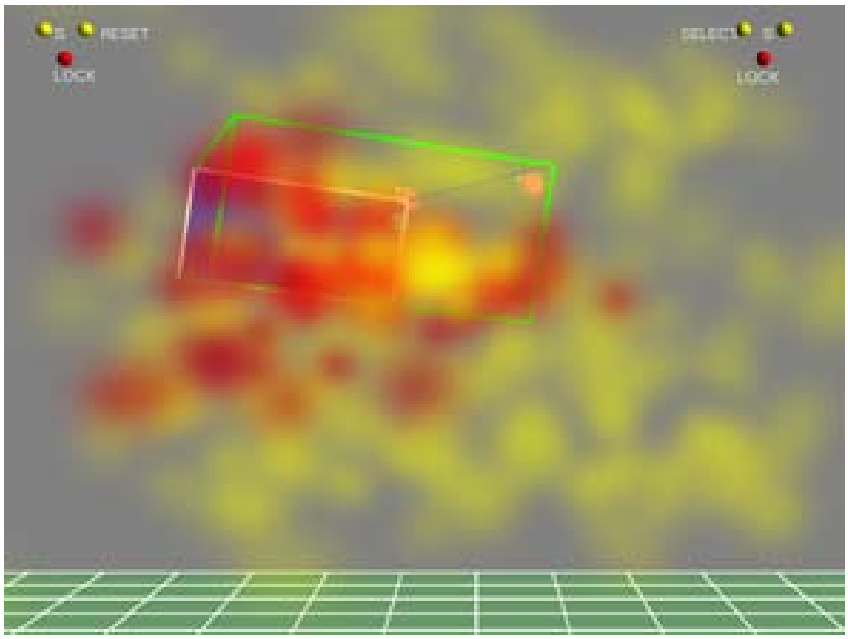


FIGURE 2.16 – Sélection de [Ulinski *et al.* 2007]

doit ensuite pointer avec le smartphone dans la direction de sélection, qui créera une ellipsoïde centrée sur le point le plus proche dans la direction pointée (figure 2.17b). Enfin, il peut ajuster la forme de la sélection en orientant le smartphone selon l'axe à modifier, puis en changeant la taille de l'ellipsoïde en fonction de l'éloignement de deux doigts sur son écran (figure 2.17c).

Utiliser des formes prédéfinies telles que vues dans cette section offre plus de contrôle sur le volume de sélection (surtout en profondeur) tout en restant faciles à manipuler, mais elles restent simples. De plus, du fait de leur forme globalement toujours identique, les formes prédéfinies ne sont pas adaptées à sélectionner n'importe quel volume et auront du mal à avoir de bonnes performances en terme de temps nécessaire pour réaliser une sélection et en terme de précision sur des formes irrégulières.

2.3.4 Objets de forme libre

La solution semble alors les objets de forme libre. Leur forme n'étant aucunement limitée, ces objets permettent en théorie d'obtenir la meilleur précision possible. Plusieurs approches existent permettant de les construire. La première est d'utiliser un logiciel de modélisation 3D, comme Blender [Blender Founda-

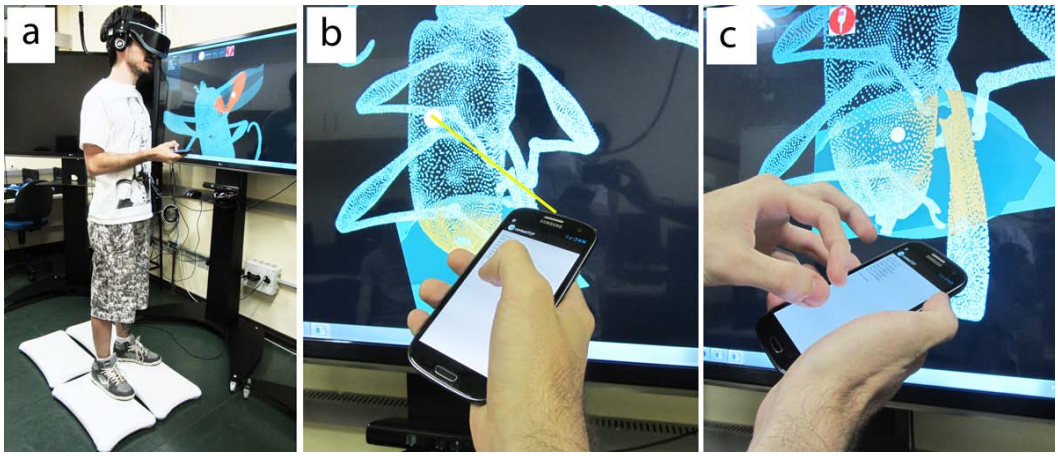


FIGURE 2.17 – Sélection de [Krammes *et al.* 2014]

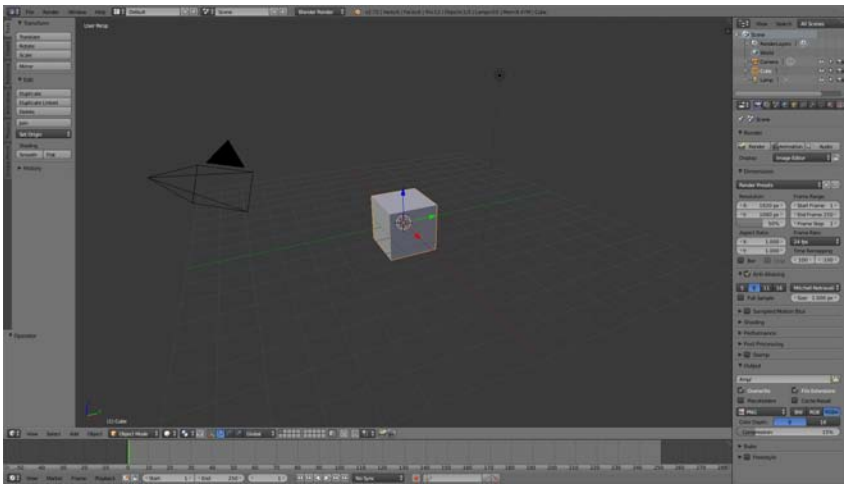


FIGURE 2.18 – Interface de Blender

tion 1995]. Cependant l'utilisation de tels logiciels est basée sur une interface complexe (figure 2.18), qui nécessite un long temps d'apprentissage avant de pouvoir créer un objet de forme libre rapidement. Un autre exemple de construction de ces objets est donné par [Kang *et al.* 2013]. À partir d'une caméra, ils détectent la position des mains d'un utilisateur ainsi que chacun de ses doigts. Ils ont ensuite créé un système permettant de réaliser des actions (dans le but de créer un objet de forme libre) en fonction de la position des doigts des deux mains. Par exemple, la translation de l'objet est réalisée en mettant une main complètement droite, doigts écartés. Cependant leur façon de faire reste compliquée à utiliser, étant donné que plus de vingt gestes sont à apprendre.

Un moyen d'utiliser les objets de forme libre tout en restant simple d'utilisa-

tion est de construire des objets contraints. Cela consiste à contraindre volontairement les formes possibles dans le but de simplifier leur utilisation. Par exemple, une forme peut être limitée à une sphère, limitée à un changement de rayon seulement, alors que plus de liberté pourrait être offerte en utilisant un ovoïde. L'équilibre recherché est alors de trouver des limitations peu contraignantes mais simplifiant grandement la construction de la forme.

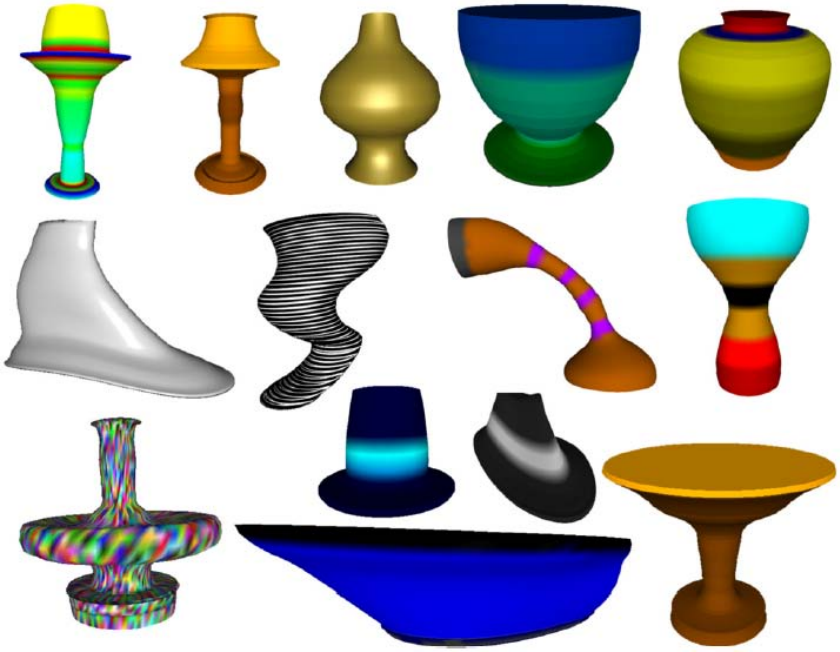


FIGURE 2.19 – Exemple de construction par [Vinayak *et al.* 2013]

La technique créée par [Vinayak *et al.* 2013] par exemple est beaucoup plus simple d'utilisation que la technique de [Kang *et al.* 2013], la forme étant limitée à un cylindre généralisé. Le principe est le même, il s'agit de créer un objet 3D à l'aide de gestes des mains captés via une caméra. Cependant seulement trois gestes sont à apprendre grâce à la contrainte ajoutée. Un exemple de formes créées à l'aide de cette technique est visible figure 2.19.

[Sowell *et al.* 2009] utilisent la construction d'un objet contraint afin de réaliser une sélection. La construction de cet objet passe par le dessin de lignes formant le contour global de l'objet, qui est construit en interpolant la surface ainsi créée. Un exemple de leur application est visible figure 2.20. Les lignes de contours sont visibles par différentes couleurs, et l'objet reconstruit est affiché en rouge. C'est une technique qui fonctionne bien mais les auteurs se limitent à la

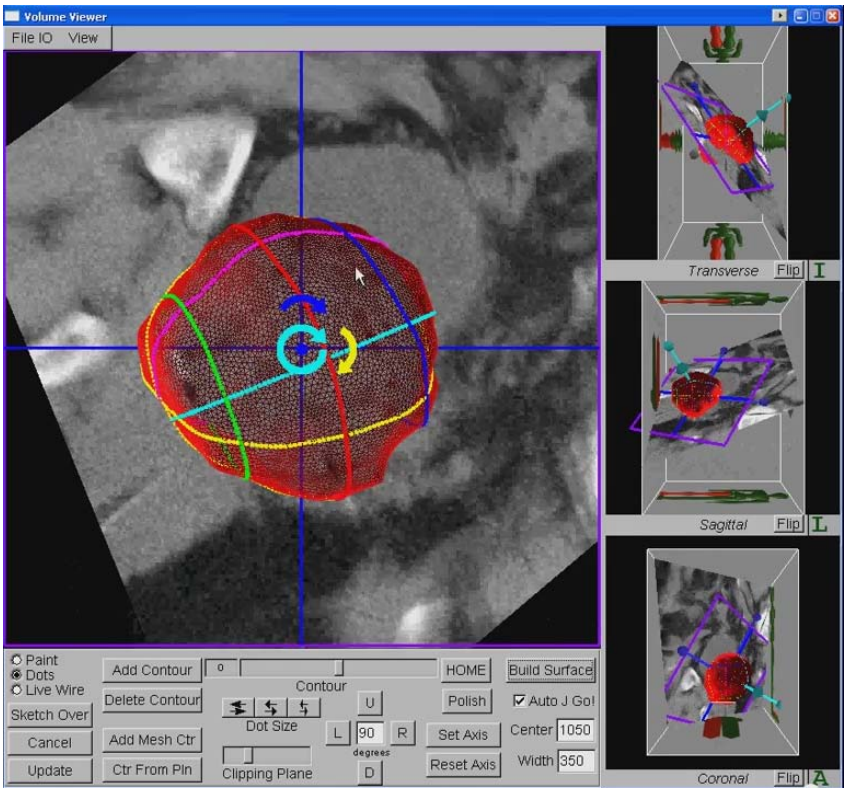


FIGURE 2.20 – Sélection de [Sowell *et al.* 2009]

prise en charge de voxels, une représentation courante dans le milieu médical.

La sélection par volume est donc un compromis entre facilité d'utilisation, flexibilité et précision offerte. D'un coté, les techniques utilisant une projection (basées sur le lancer de rayon, qui consiste à envoyer un rayon dans la scène 3D à partir du curseur pour retrouver le modèle poiinté) sont très simples à utiliser. Elles offrent une certaines flexibilité mais ne permettent pas une grande précision pour sélectionner des points. De l'autre coté, les objets de forme libre ont le meilleur potentiel de précision mais leur utilisation est complexe à cause de multiples degrés de liberté permis. Entre les deux se trouvent les formes prédéfinies qui sont simples à manipuler et adaptables, mais seulement jusqu'à un certain point (changement de taille principalement).

2.4 Traitement de modèle

Cette thèse porte un intérêt particulier aux nuages de points. C'est cette représentation que nous avons choisie de traiter dans nos exemples, et celle qui est disponible via le projet encadrant cette thèse (voir section 1.2). Mais les nuages de points viennent couramment des acquisitions avec des scanners lasers, et possèdent des problèmes. Il est donc en général obligatoire de les faire passer par différents traitements avant d'être utilisables par leur application cible, comme la visualisation par exemple.

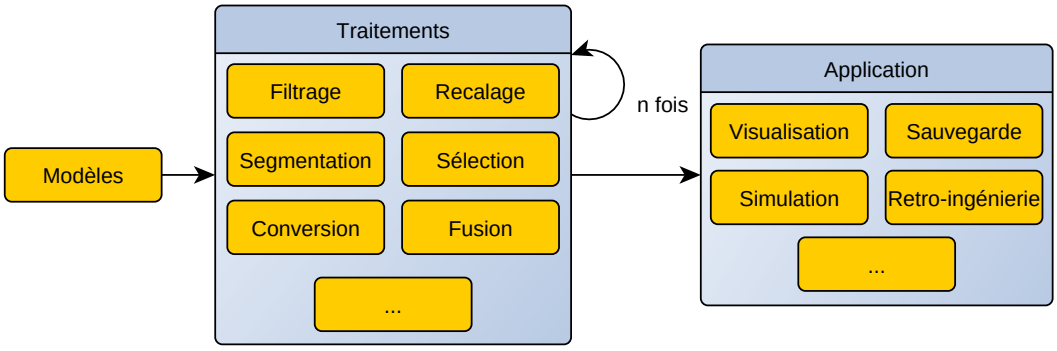


FIGURE 2.21 – Traitements et applications possibles

Un ensemble non-exhaustif de ces traitements est visible sur la figure 2.21. Elle commence par l'acquisition du modèle, suivi d'une succession de traitement dont l'ordre et la nature vont varier selon les applications. Par exemple, si un modèle acquis est fait pour être affiché dans un simulateur d'aviation, on peut raisonnablement penser que ce modèle ne sera vu que de loin, et ne nécessitera donc pas une résolution élevée, ce qui aidera également le moteur graphique car de nombreux objets seront affichés. Les traitements seront alors un filtre de simplification, suivi d'une conversion en un maillage, et enfin d'un recalage dans l'environnement 3D de la simulation. Les traitements visibles sont détaillés un peu plus par les sections suivantes.

La figure 2.21 présente enfin un ensemble d'applications possibles. La visualisation permet simplement l'affichage du modèle 3D, ainsi que la navigation autour et à l'intérieur. La simulation quant à elle va aller un peu plus loin et va permettre l'animation du modèle, comme une ouverture de porte, mais aussi, après rétro-ingénierie, le calcul de contraintes pour par exemple tester si un toit peut résister à la pression d'une chute de grêle. La rétro-ingénierie justement a pour but de

retrouver précisément l'objet représenté par le modèle, mettre une étiquette sur chacun de ses composants, peut-être également les matériaux qui le composent. Enfin, la sauvegarde est possible pour préserver numériquement un patrimoine, ou garder un modèle de référence dans le cas de dégradation de l'objet réel, par exemple dans le cas d'un incendie.

2.4.1 Filtrage d'un nuage

Le principe du filtrage est de passer l'ensemble des points d'un nuage dans un algorithme pour n'en garder qu'une sous-partie, selon un critère dépendant du type de résultat voulu. Nous donnons ci-après deux exemples de ces traitements.

2.4.1.1 Suppression du bruit

Lorsqu'un nuage de points est obtenu à l'aide d'un scanner laser, le résultat n'est pas parfait. À cause de différents problèmes, certains points sont faux. Ils ne représentent pas la surface de l'objet scanné. Par exemple, le scanner possède une précision limitée, de fait si l'objet scanné est trop éloigné le risque d'obtenir des points erronés augmente. Le but de ce filtre est alors de détecter les points concernés et de les supprimer du nuage.

On parle régulièrement en termes de fréquence. De la même manière que dans une image, les basses fréquences vont dès lors représenter les zones planes et les hautes fréquences les zones à grande variation de couleur. On déduira alors qu'une grande variation au milieu d'une zone plane n'a pas lieu d'être, et sera considérée comme du bruit.

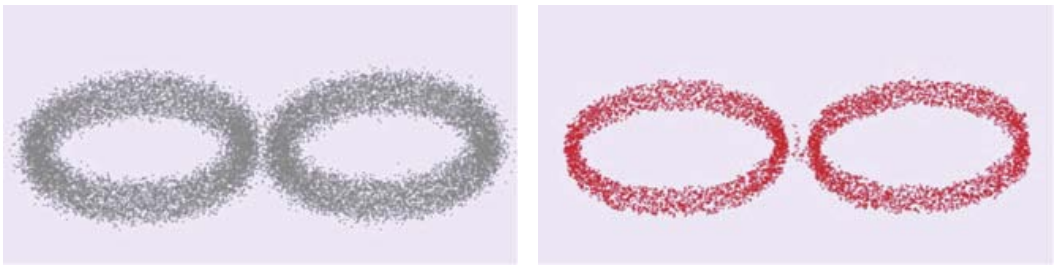


FIGURE 2.22 – Suppression du bruit par [Hou *et al.* 2012]

[Hou *et al.* 2012] ont choisi d'utiliser une méthode assez simple mais effective. Ils commencent par lisser le nuage de points, à la manière d'un filtre ne laissant

passer que les faibles fréquences (les hautes fréquences représentant les plus grand écarts et donc le bruit). En fonction de la manière dont chaque point est déplacé, des seuils sont ensuite automatiquement choisis. Enfin, tous les points ayant réalisé un déplacement plus grand que ce seuil sont considérés comme du bruit et supprimés. Le principe peut ensuite être répété plusieurs fois jusqu'à obtenir un résultat satisfaisant. La figure 2.22 montre le résultat d'une passe de leur algorithme.

FIGURE 2.23 – Suppression du bruit par [Digne 2012]

[Digne 2012] a choisi une solution différente pour s'attaquer à ce problème. Elle cherche également a ne retirer que les hautes fréquences mais calcule pour cela un descripteur local dépendant du point à analyser et de ses voisins. Le calcul de ce descripteur permet d'obtenir un vecteur représentant la surface locale de chaque point. Elle applique enfin un critère de similarité pour détecter le bruit et le supprimer ; si le vecteur d'un point est trop différent de ses voisins, il est supprimé. Un résultat obtenu à l'aide de son algorithme est visible figure 2.23.

Enfin, [Park *et al.* 2013] ont adopté encore une autre approche pour supprimer le bruit. Ils commencent pour cela par calculer la normale de chaque point de façon robuste, en détectant si un point fait partie plutôt d'une surface plane ou d'un coin. Ensuite, ils utilisent simplement la normale afin de réaliser leur filtre. Si la normale d'un point est trop différente de ses voisins alors qu'il ne fait pas partie d'un coin, cela signifie que ce point est du bruit. Un exemple de résultat de leur algorithme est visible sur la figure 2.24.

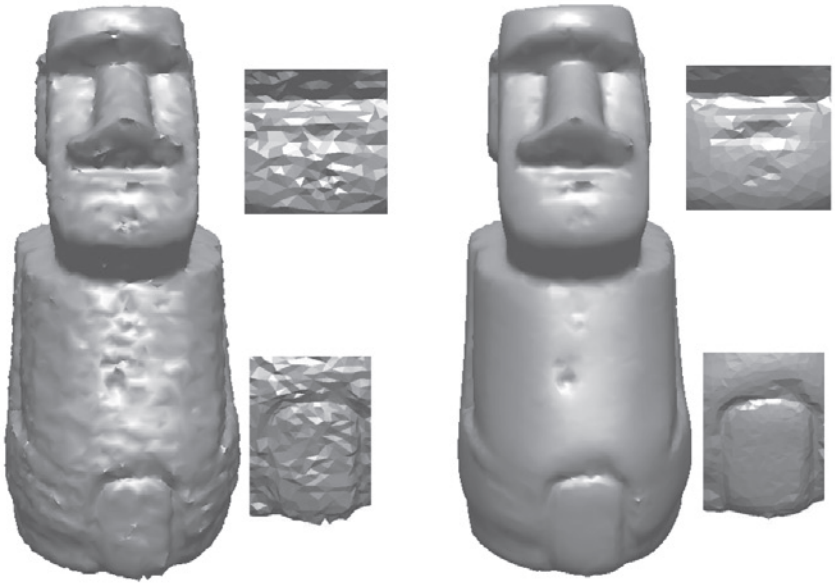


FIGURE 2.24 – Suppression du bruit par [Park *et al.* 2013]

2.4.1.2 Simplification

Le nombre de points obtenus à l'aide des scanners laser est en général important. Il peut facilement dépasser la centaine de million. Cela est voulu car la précision ainsi apportée permet de retrouver des détails très fins. Cependant ce nombre important de points peut rendre la manipulation du modèle 3D difficile. En effet, cela nécessite des algorithmes adaptés et des ordinateurs suffisamment puissants. Or toutes les applications n'ont pas forcément besoin d'un niveau de détail si important. Il peut également être utile d'appliquer un algorithme sur un modèle simplifié puis d'appliquer ses résultats sur le modèle complet pour gagner du temps. Pour réaliser cela on utilise des algorithmes de simplification, qui permettent de réduire le nombre de points d'un modèle sans en perdre l'apparence générale.

[Pauly *et al.* 2002] ont réalisé une revue des différentes méthodes utilisées pour réaliser une simplification. Ils les ont classées en trois groupes :

— la méthode par regroupement, qui divise un nuage de points en plusieurs sous-parties, remplaçant chacune de ces sous-parties par un unique point représentatif [Brodsky & Watson 2000] ;

— la méthode itérative, qui cherche à effondrer des paires de points selon une métrique jusqu'à obtenir le nombre de points désiré [Alexa *et al.* 2001] ;

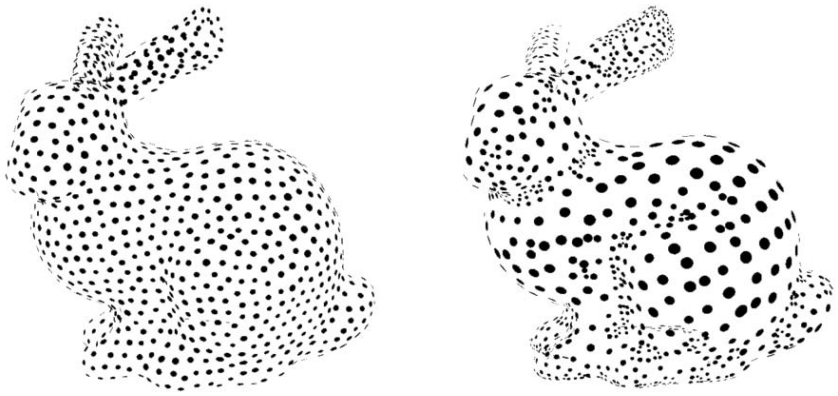


FIGURE 2.25 – Simplification par [Pauly *et al.* 2002]

— la méthode par simulation de particule, qui va ré-échantillonner la surface du nuage de points selon des forces inter-particule repoussantes [Turk 1992].

Un exemple de leur implémentation de la première méthode est visible figure 2.25.

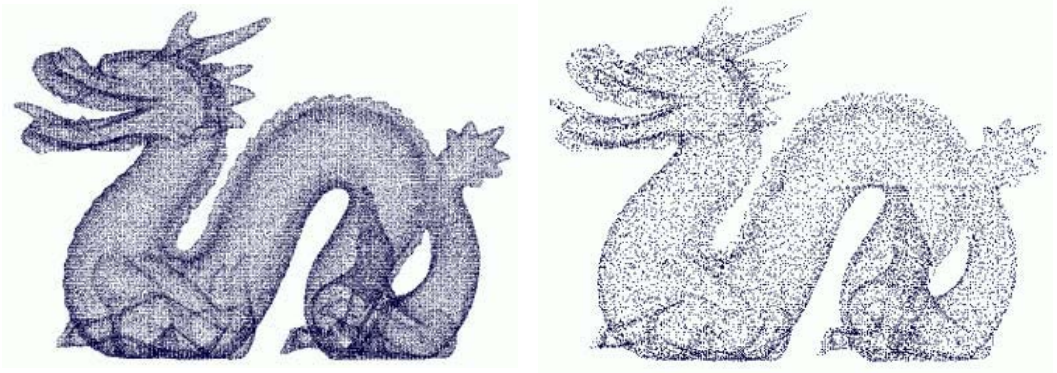


FIGURE 2.26 – Simplification par [Wang *et al.* 2007]

[Wang *et al.* 2007] dans une publication plus récente ont utilisé une nouvelle méthode. Ils commencent par convertir les points du nuage en une image de géométrie en projetant leurs coordonnées polaires dans un plan, ce qui leur permet par la suite de calculer rapidement leur courbure et leurs points voisins. Enfin ils calculent la densité locale et utilisent ce paramètre associé à la courbure pour simplifier la surface. Un exemple de leur résultat est visible figure 2.26.

Une autre approche permettant la simplification est le ré-échantillonnage. Cela permet de replacer des points repartis uniformément sur la surface à la densité voulue. Ce traitement est aussi utile dans d'autres cas comme la suppression de bruit, but de [Huang *et al.* 2013]. Afin de conserver les détails du modèle, ils

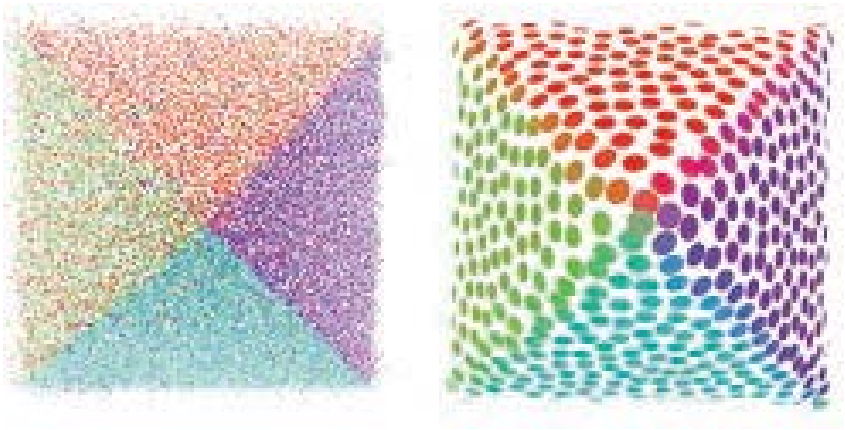


FIGURE 2.27 – Simplification par [Huang *et al.* 2013]

ont cherché à faire un ré-échantillonnage qui préserve les bords de façon nette. Pour cela ils créent un nouveau modèle et commencent d'abord par lui ajouter des points dans les zones planes, puis s'approchent petit à petit des bords lorsqu'ils sont de plus en plus sûrs de la surface représentée. Un exemple de leur résultat est visible figure 2.27.

2.4.2 Recalage

Le but du recalage est de placer un ensemble de modèles dans le même repère. En effet, dans un modèle, l'information qui a du sens est la position relative de ses points par rapports aux autres. Modifier l'origine du modèle ne modifie pas l'objet représenté, mais il est obligatoire d'en choisir une pour pouvoir l'enregistrer numériquement. La même chose est vraie pour l'orientation et l'échelle. Si l'unité choisie est le mètre, l'objet ne changera pas en changeant l'unité en millimètres.

Cependant cela pose un problème lorsque l'on possède plusieurs modèles, leur repère (origine, orientation et échelle) n'étant pas le même, ce qui pose problème pour les comparer ou pour les intégrer dans un même environnement 3D. Le recalage sert donc à analyser un ensemble de modèles, à choisir arbitrairement un repère commun et à modifier l'ensemble des points des modèles pour refléter ce changement.

Afin de pouvoir être recalés automatiquement ensemble, chaque modèle doit avoir une partie commune avec un autre. Un algorithme de recalage trouvera alors ces parties communes et en déterminera la transformation à appliquer pour

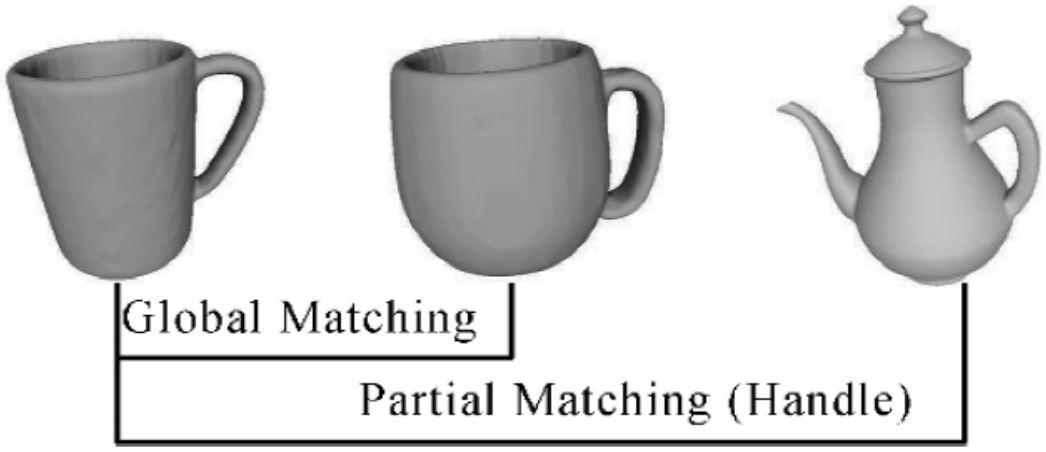


FIGURE 2.28 – Différence entre local et global de [Liu *et al.* 2013]

placer les modèles dans le même repère. Selon [Liu *et al.* 2013], les algorithmes de reconnaissance de partie commune peuvent être divisés en deux : les méthodes globales et les méthodes locales (figure 2.28). La différence entre les deux se fait par les zones de recherches ; la méthode globale ne travaillera que sur la totalité des modèles, alors que la méthode locale cherchera à faire correspondre des sous-parties, ce qui permet de recaler des modèles partiels. C'est donc cette seconde approche qui nous intéresse le plus.

Toujours selon [Liu *et al.* 2013], la méthode locale peut encore être divisée en trois sous-catégories. La première utilise des descripteurs locaux, des vecteurs qui représentent la géométrie locale (courbure...). Deux parties sont alors communes lorsqu'elles partagent un ensemble de descripteurs locaux. La deuxième se base sur une segmentation (voir section suivante) faite sur les modèles. Elle a l'avantage de donner des résultats plus proches de ce qu'un humain pourrait faire, mais souffre de défaut de jeunesse (comme devoir choisir le nombre de partie pour la segmentation). Enfin, la troisième méthode se fait par vue. Un ensemble d'image des modèles est généré selon plusieurs points de vue, et la reconnaissance se fait ensuite strictement à partir de ces images. Cette méthode a l'avantage de pouvoir effectuer des reconnaissances de modèles 3D avec des images ou des cartes de profondeur.

2.4.3 Segmentation et sélection

La segmentation et la sélection ont un but commun : extraire des sous-parties d'un modèle. Là où ils différent, c'est que la segmentation sera un algorithme appliqué par un ordinateur et fournira un ensemble de sous-parties, alors que la sélection sera réalisée par un humain assisté par l'ordinateur dans le but d'extraire une unique sous-partie d'intérêt. Ces traitements ont été expliqués plus en détail dans la section 2.3.

2.4.4 Conversion

La conversion a pour but de changer la représentation d'un modèle 3D. Typiquement, toute conversion d'un nuage de points commencera par retrouver la surface représentée. Une fois ceci fait, la surface sera convertie dans la représentation de son choix.

Dans le cadre de la conversion en maillage, [Labatut 2009] détermine trois méthodes. La première consiste à convertir le nuage de points en une fonction implicite, c'est-à-dire une fonction qui prend une position dans l'espace et renvoie une valeur. Une valeur arbitraire est alors choisie qui définit la surface, avec toutes les positions qui renvoient une valeur plus faible définissant l'intérieur du modèle, et une valeur plus forte comme l'extérieur. Cette fonction est ensuite convertie en maillage, avec par exemple la méthode des *marching-cube*. Cette méthode a l'avantage de toujours donner une surface lisse et sans trous, sans défauts. La deuxième se base sur la triangulation de Delaunay. Cette méthode suppose une surface densément peuplée et sans bruit, et triangule ainsi l'ensemble des points. Les triangles en trop sont alors supprimés selon un critère prédéfini et le maillage est obtenu. Enfin, la troisième méthode se base sur un modèle déformable. Le modèle va aller s'adapter petit à petit au nuage de points, en fonction d'un calcul d'énergies. Cette méthode a pour désavantage d'être sensible à l'initialisation et aux minimas locaux.

2.4.5 Fusion

La fusion a pour but de réunir deux modèles en un seul. Dans le cas d'un nuage de points, ce n'est pas très compliqué, il suffit d'enregistrer les deux modèles en-

semble. Certains traitements peuvent ensuite être effectués pour obtenir un modèle de meilleure qualité, comme supprimer les éventuels doublons ou ajuster la densité du nuage de points pour qu'elle soit uniforme.

Dans le cadre de maillage la fusion est plus compliquée. En effet, il faut être capable de construire une nouvelle surface pour le nouveau modèle là où les deux modèles sources s'entrecroisent. Plusieurs travaux proposent des solutions pour cela, comme [Kanai *et al.* 1999, Jin *et al.* 2006].

2.4.6 Remplissage de trous

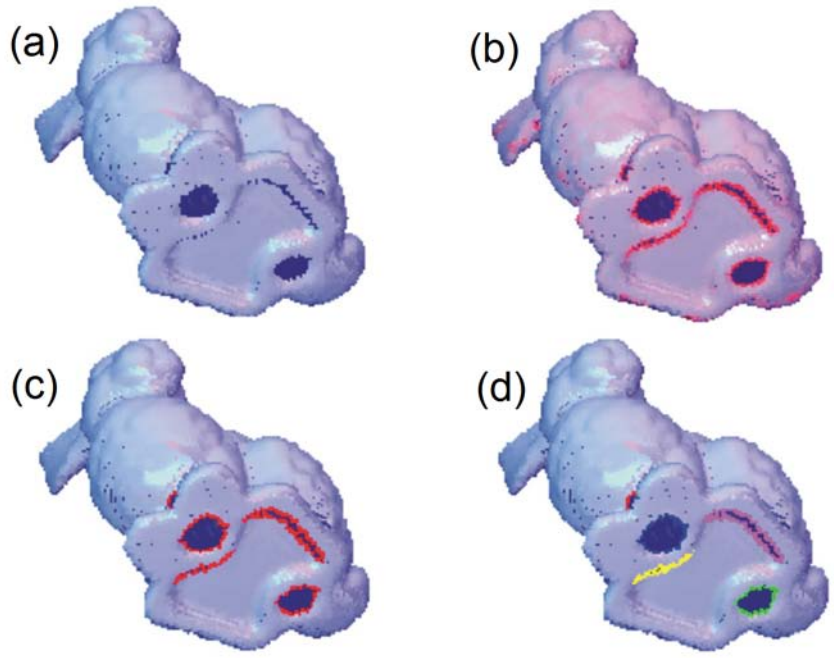
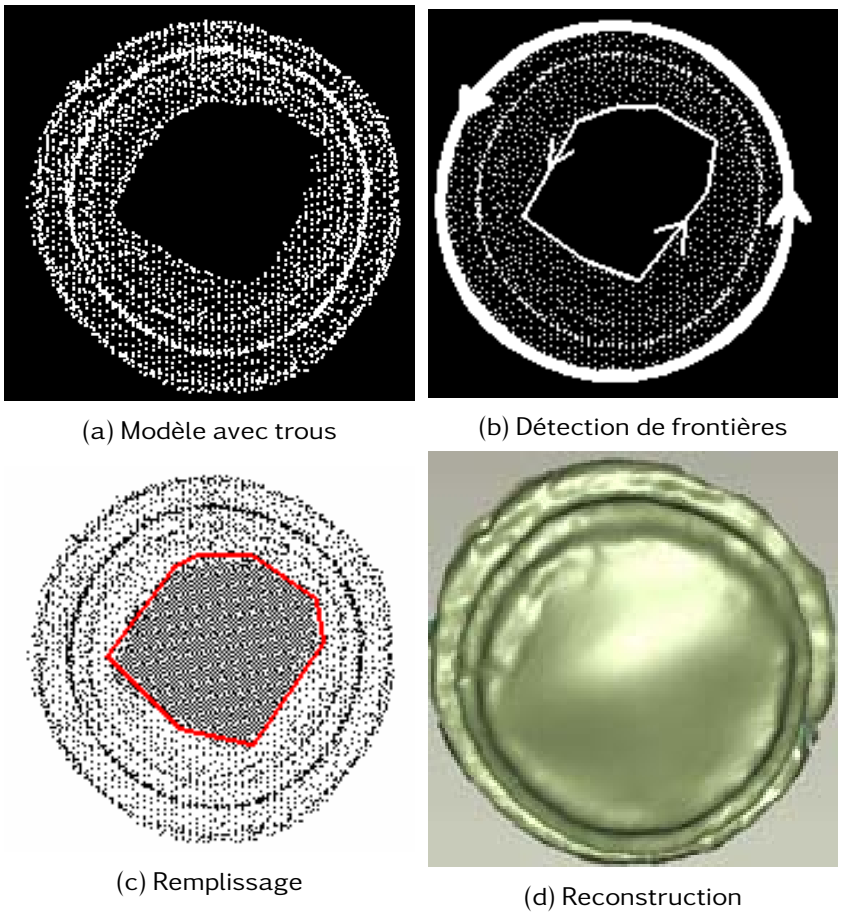


FIGURE 2.29 – Détection de trou par [Bendels *et al.* 2006]

Le remplissage de trous est le traitement qui permet de trouver des trous dans le nuage de points puis de déterminer si ces trous existent dans l'objet représenté ou non. Dans ce dernier cas, le but est de remplir le trou en rajoutant de nouveaux points de façon à ne pas remarquer qu'un trou existait avant à cet endroit. Les nuages de points acquis via un scanner souffrent toujours de trous, à cause d'occlusion au moment de l'acquisition. [Kirkvik 2008]dans son rapport de master parle plus en détails du problème et fait une revue de solutions possibles. Par contre l'auteure a choisi d'appliquer ses traitements sur des nuages

de points reconstruits, transformés en maillages donc. Remplir les trous d'un maillage est un pré-traitement fréquent ayant fait l'objet de nombreux travaux [Pernot *et al.* 2006]. La raison de ce choix est probablement indiquée par [Bendels *et al.* 2006] : la détection de trous dans les nuages de points est difficile. En effet, il ne s'agit pas simplement de trouver des zones vides entourées de points, car la surface de l'objet représenté est inconnue. Par exemple, il est normal d'avoir un trou dans l'anse d'une tasse.



(a) Modèle avec trous (b) Détection de frontières

(c) Remplissage (d) Reconstruction

FIGURE 2.30 – Remplissage de trou par [He & Cheng 2011]

Pour détecter les trous, [Bendels *et al.* 2006] procèdent en plusieurs étapes. En partant d'un modèle exemple (figure 2.29a), les auteurs calculent pour chaque point sa probabilité d'être placé sur une frontière (les points rouges sur la figure 2.29b). Les points trouvés sont ensuite classés en frontière ou point intérieur (figure 2.29c). Enfin, chaque frontière est suivie pour trouver tous les points qui la composent (colorées d'une couleur différente sur la figure 2.29d), trouvant ainsi

les trous.

FIGURE 2.31 – Remplissage de trou par [Doria & Radke 2012]

Une fois qu'un trou a été détecté, les difficultés continuent. Il s'agit de le boucher de manière cohérente avec le reste du modèle. [He & Cheng 2011] utilisent par exemple une approche similaire pour détecter les trous, en commençant par rechercher les points appartenant à une frontière. Partant d'un modèle comportant des trous (figure 2.30a), ils se basent sur un *octree* leur permettant de rechercher les plus proches voisins de chaque point, puis calculent le plan qui leur correspond. À l'aide de ces plans, les frontières sont détectées (figure 2.30b). Ils trouvent les frontières intérieures (les trous) puis les remplissent selon l'axe des plans qui les composent (figure 2.30c). Ils rajoutent enfin une étape de reconstruction (figure 2.30d).

[Doria & Radke 2012] travaillent quant à eux sur des nuages de points un peu plus particuliers car il s'agit d'images de profondeur avec couleur. Ils commencent par détecter les trous à l'aide des différences de profondeur entre chaque point. Puis ils génèrent des nouveaux points à la profondeur désirée et calculent leur couleur par des techniques de remplissage de trous dans des images (*inpainting*). Un de leur résultat est visible figure 2.31.

2.5 Synthèse

Ce chapitre a d'abord commencé par rappeler les bases des environnements 3D. Après en avoir donné la définition, ce chapitre a décrit les différents types de

modèles qu'il est possible de retrouver dans un environnement virtuel ainsi que les interactions possibles avec ceux-ci.

Nous sommes ensuite passés sur le traitement de modèles en nuage de points, qui est nécessaire pour leur utilisation. Nous avons remarqué qu'ils sont très nombreux, en donnant des exemples parmi les plus utilisés, mais aussi que choisir les traitements à effectuer est dépendant de l'application voulue. Un traitement qui propose de corriger un ensemble de problèmes en une seule fois devient alors utile, et c'est l'une des contributions de cette thèse comme vous allez le voir dans le chapitre suivant.

Chapitre 3

Première contribution : un processus d'aide à la conception d'environnements 3D

Le chapitre précédent a posé les travaux antérieurs pour donner une vision claire des domaines abordés. Ce chapitre va décrire plus en détails le contexte de travail dans lequel cette thèse place ses contributions et apporter les éléments qui montrent que des problèmes existent. Il se terminera par un aperçu de la solution proposée, notre processus en quatre étapes, qui sera détaillée dans la partie suivante.

3.1 Solution proposée

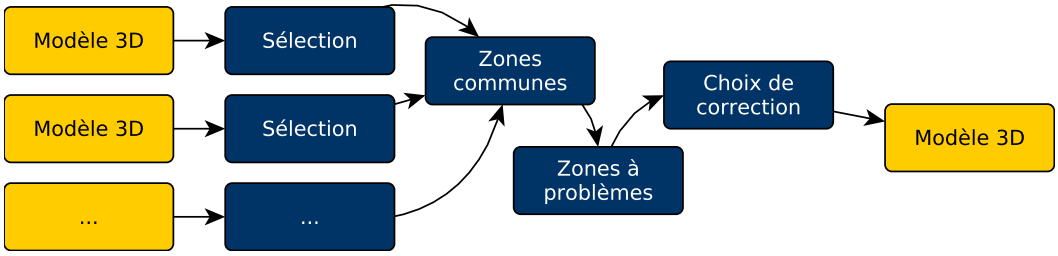


Figure 3.1 – Le processus proposé

Le principe est donc de tirer partie de la multiplicité des modèles 3D représentant un même objet, dans le but d'obtenir un modèle final amélioré [Panchetti *et al.* 2010]. Cependant la multiplicité des représentations et des sources disponibles ne rend pas la chose aisée. En effet, ces modèles vont contenir des informations se recoupant, d'autres non, à des résolutions différentes, et le but est de les regrouper ensemble. Cela se fait donc en plusieurs étapes, qui seront rapidement

décrites avant de les détailler plus dans la partie suivante. Le processus présenté ici vise à fonctionner pour n'importe quelle représentation de n'importe quel objet.

Cependant il n'a été implémenté et testé que dans le cadre de l'amélioration d'un nuage de points à l'aide de maillages.

Le processus développé pour cet objectif est visible sur la figure 3.1. Il comporte quatre étapes, mais a un prérequis important : tous les modèles en entrée doivent se trouver dans le même repère. C'est-à-dire qu'ils utilisent la même unité et possèdent la même origine. Une autre façon de voir les choses est que si on choisissait de les afficher tous dans la même application sans traitements, ils seraient tous affichés en superposition. Si tel n'est pas le cas, les modèles ont besoin d'être soumis à un algorithme de recalage.

3.1.1 Sélection

La première étape proposée dans notre processus est la sélection. En effet, différentes représentations ne contiennent pas exactement la même information. Une partie de l'information peut ne pas être pertinente dans le cas choisi. La garder consommerait alors inutilement des ressources, ce qui peut avoir un effet non négligeable si l'information est précise. Un autre cas d'utilisation pourrait être de volontairement ignorer des zones du modèle. Il est en effet possible de ne pas vouloir appliquer le traitement sur le modèle complet.

Dans tous les cas, le principe est de choisir quelle zone sera traitée par les étapes suivantes.

3.1.2 Détection de zones communes

Une fois que les zones de traitements ont été choisies, l'étape de détection de zones communes vise à connaître plus précisément les informations disponibles. Cette étape permet de détecter quelles sont les zones où un recouvrement existe, de quels modèles et à quelle résolution. Cela permet également de déterminer les zones qui n'existent que dans un seul modèle source.

3.1.3 Détection de zones à problème

Ensuite, en s'appuyant sur ces zones, le processus vise à détecter les problèmes qui peuvent exister dans chacune d'entre elles. Peut être que la résolution n'est pas suffisante et peut-être améliorée via un autre modèle, ou peut-être que le problème est qu'une zone n'existe que dans un seul modèle. Les problèmes ainsi détectés dépendent fortement des représentations sources ainsi que de la représentation cible choisie.

3.1.4 Résolution des zones à problème

Enfin, la dernière étape consiste à corriger les problèmes ainsi détectés et générer le modèle corrigé et complété. La qualité du modèle résultant dépendra beaucoup des modèles d'entrée et du but recherché. En effet, avoir une résolution élevée par exemple n'est pas forcément utile ou idéale pour certaines applications.

3.2 Développement de l'environnement de travail

3.2.1 Base de développement

Un pré-requis important mais simple à ces travaux était de pouvoir manipuler les différents modèles 3D dans un même outil de visualisation afin de réaliser les traitements nécessaires, pour la sélection d'une part et l'amélioration d'autre part. Il était également important de pouvoir gérer la mémoire facilement, car un modèle tel celui du Pic du Midi est gros et contient trop de points pour les afficher sans gestion particulière.

Nous avons donc commencé par la conception et le développement d'un rapide et simple moteur 3D supportant nos besoins. Cela nous a permis d'avoir le contrôle total de la gestion des modèles 3D, et de rester flexible, pour nous adapter à toutes les modifications nécessaires facilement quand le besoin s'en ressentait. En échange cela a probablement pris un peu plus de temps qu'utiliser une bibliothèque déjà existante.

Le moteur graphique a donc été basé sur un système classique en graphe de

scène, chaque nœud représentant une partie de la scène, comme par exemple un modèle 3D, une transformation ou une caméra.

L'application permet donc de charger des modèles 3D composés de points ou de triangles, et de les afficher de plusieurs manières (en jouant sur les couleurs). Elle permet également de traiter ces modèles avec notre processus. Enfin, elle a servi de base pour développer tout ce dont nous avons eu besoin pendant la thèse, comme le prototype de sélection (voir chapitre 4).

3.2.2 Extraits de mise en œuvre dans la base

3.2.2.1 Système de signal

Le graphe dans l'implémentation réalisée est géré par un gestionnaire, une classe qui va gérer la mise en cache de certaines opérations pour plus de rapidité. Elle permet aussi la ré-utilisation de nœuds à plusieurs endroits, par exemple quand un élément est affiché plusieurs fois. Il fallait donc un moyen de communication entre le gestionnaire et les nœuds, sans que ceux-ci se connaissent explicitement.

Un système de signal a donc été mis en place. Quand un nœud est rajouté, par exemple, le gestionnaire va écouter ses signaux pour réagir de manière appropriée. Ici, il va donc falloir reconstruire le cache car le graphe a changé. Cela peut aussi arriver quand un nœud fait une action qui ne change pas le graphe mais change le rendu graphique.

Le système a été conçu pour être très simple. Il consiste en un objet qui va conserver des fonctions. Ces fonctions seront données par les objets qui veulent réaliser une action lorsqu'un signal est émit. Le principe sera donc, lorsqu'un signal est émis, d'appeler toutes les fonctions qui ont été données, une par une. Les signaux peuvent ainsi véhiculer une information de par leur nature, mais également par des paramètres qu'ils sont capables de transmettre.

Le système est également capable de modularité. Ainsi il est possible d'écouter, selon son choix, les signaux de l'ensemble des nœuds existants, ou alors de seulement ceux présents dans un graphe particulier, ou encore seulement de ceux contenant une représentation graphique d'un objet, ou enfin uniquement d'un seul nœud.

3.2.2.2 Mémoire

Afin de pouvoir traiter des données trop larges pour être stockées entièrement dans la mémoire vive simplement et facilement, un gestionnaire de mémoire spécial a été créé. Le principe est d'allouer la mémoire par blocs et de décharger les blocs non utilisés de manière transparente pour l'utilisateur, ainsi que de les recharger en mémoire de la même manière lorsqu'il sont nécessaires.

La première étape est de définir la taille des blocs et la taille maximale utilisable. Ensuite, pour suivre la mémoire utilisée, une liste globale de blocs actuellement chargés en mémoire est maintenue, mais de manière complètement invisible pour un utilisateur. À chaque nouveau besoin en mémoire, cette liste sera vérifiée pour respecter les contraintes. Pour qu'un utilisateur puisse avoir accès à cette mémoire, un type équivalent à un vecteur C++ (capable de s'agrandir) est fourni.

Lorsqu'un nouveau vecteur est créé, qu'elle que soit sa taille, il ne fait qu'allouer le nombre de blocs nécessaire pour stocker ses données mais la mémoire pour les données elle-même n'est pas allouée. Cela permet de créer des vecteurs de taille arbitraire, quelque soit la mémoire vive disponible. Chaque bloc est alloué uniquement lorsqu'il est utilisé pour la première fois, ou lorsqu'il a été déchargé de la mémoire. Dans ce dernier cas, la mémoire est rechargée depuis le système de fichiers avant d'en permettre l'utilisation.

Un aperçu du fonctionnement de l'accès à la mémoire d'un bloc est détaillé par l'algorithme 1. La liste gardant les blocs alloués permet également de choisir quel bloc il faut libérer pour ne pas dépasser la limite. Le fonctionnement montré ici va libérer le bloc qui a été le plus anciennement utilisé. L'algorithme présente ici une version simplifiée. Par exemple, dans l'implémentation utilisée, le bloc ne sera pas sauvegardé dans un fichier à sa libération s'il n'a pas été modifié.

Cette technique permet donc d'utiliser des vecteurs de taille arbitraire de façon transparente et simple, au prix d'une dégradation des performances.

3.2.2.3 Mémoire graphique

Afin de visualiser les traitements réalisés en temps réel, il est important de synchroniser les données de la mémoire graphique, qui contient tout ce qui est

Algorithme 1 Accès à la mémoire d'un bloc

```
fonction accesMemoireBloc(bloc, memoireAllouee, maxMemoire, listeBlocEnMemoire)
    si bloc n'est pas déjà en mémoire alors
        si bloc.taille + memoireAllouee est plus grand que maxMemoire alors
            dernierBloc ← dernier bloc de listeBlocEnMemoire
            sauvegarder mémoire dans fichier de dernierBloc
            libérer mémoire de dernierBloc
            enlever dernierBloc de listeBlocEnMemoire
        fin si
        allouer mémoire de bloc
        si fichier existe pour bloc alors
            copier fichier dans mémoire allouée de bloc
        fin si
    fin si
    mettre bloc en première position de listeBlocEnMemoire
    retourner mémoire de bloc
fin fonction
```

nécessaire à la création de l'image finale, avec celles de la mémoire vive. En effet, sans cette synchronisation, toute modification du modèle 3D, bien que réelle, ne serait pas visible. Les modèles ainsi synchronisés ne doivent évidemment pas être trop gros. La mémoire graphique étant encore plus petite que la mémoire vive, un traitement particulier est nécessaire pour afficher les modèles ayant besoin de la structure décrite à la section précédente.

La synchronisation n'est pas difficile en soi mais est facile à oublier, de même qu'elle peut être source de problèmes si elle est mal réalisée. Pour éviter tous ces problèmes, une classe particulière a été créé. Elle est prévue pour contenir le modèle 3D stocké dans la mémoire vive mais également son équivalent dans la mémoire graphique. De cette façon, la classe suit les modifications effectuées et met à jour automatiquement la mémoire graphique au moment approprié. Cette approche permet d'éviter tout oubli ou problème en complète transparence pour un développeur.

3.3 Conclusion

Comme ce chapitre l'a montré, cette thèse bénificiait d'un contexte bien défini. Il a permis d'encadrer la thèse et de diriger les travaux sur des points particulièrement intéressants et basés sur des besoins réels.

Cette thèse a donc commencé par l'établissement d'une base de travail solide, qui nous a permi ensuite de faire nos contributions sans être dérangés par les détails sous-jacents. Contributions qui sont présentées dans la suite de ce manuscrit.

Chapitre 4

Deuxième contribution : sélection dans un nuage de points

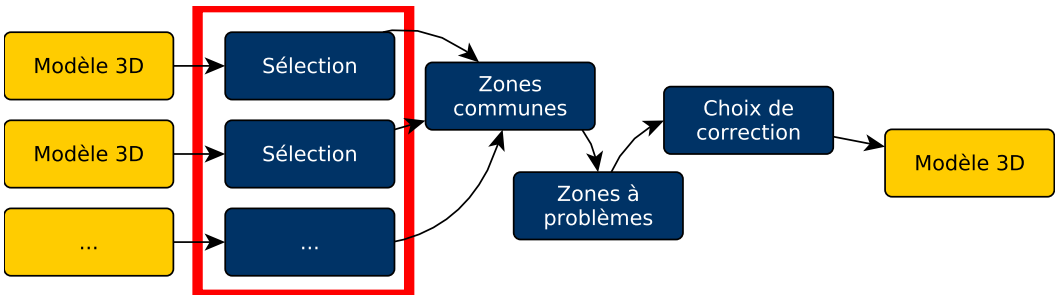


Figure 4.1 – Première partie du processus

Après avoir présenté les différents éléments de l'état de l'art concernant les domaines englobés dans cette thèse, ce chapitre détaille notre première contribution. Il s'agit d'une nouvelle approche permettant de réaliser une sélection dans un nuage de points, première étape de notre processus (entourée en rouge sur la figure 4.1). Ce chapitre détaillera l'intérêt de cette nouvelle technique par rapport à celles déjà existantes, puis la présentera plus en détails sur le principe ainsi que l'implémentation faite pour la tester. Nous passerons ensuite à la description de l'expérience utilisateur réalisée pour vérifier ses performances, et les résultats obtenus. Enfin, les perspectives de la techniques seront discutées.

Nous avons appelé cette technique Enveloppe de Sélection Adaptative (ESA). Elle a fait l'objet d'une publication [Hamelin & Dubois 2015].

4.1 Contexte

Comme nous l'avons vu précédemment, ces travaux ont été menés dans le cadre d'une collaboration avec l'équipe du Télescope Bernard Lyot (TBL) [tbl].

Leurs opérations de maintenance les amènent à intervenir pour la réparation ou l'ajout d'outils scientifiques. La planification de ces interventions s'appuie sur des plans datant de la construction de l'instrument. À l'aide d'un scanner laser, un nuage de points 3D représentant l'intérieur et l'extérieur du bâtiment contenant le télescope a été réalisé par des géomètres. Certaines opérations nécessitent l'identification de zones d'intérêt dans ce nuage de points pour valider leur faisabilité. C'est donc pour l'équipe du télescope (utilisateurs occasionnels et peu experts en 3D) que cette technique a été conçue, en nous appuyant sur l'analyse de l'activité conduite au préalable. Plus généralement face à la démocratisation des scanners 3D, leur prix et facilité de manipulation, l'accès à des nuages de points est accru même pour des utilisateurs non avertis, pour par exemple une utilisation couplée avec une imprimante 3D. Les modalités choisies pour piloter la technique de sélection doivent donc être adaptées à des utilisateurs de tous niveaux de compétence en termes de 3D et sur un poste de travail classique.

4.2 Sélection par volume

Il a été détaillé dans le chapitre 2 plusieurs manières de réaliser des sélections, que nous avons regroupé en plusieurs groupes. Le premier groupe parlait d'utiliser une segmentation automatique, ce qui pose les problèmes de trouver le bon algorithme associé à ses bons paramètres, mais qui nécessitera sans doute encore un ajustement par la suite. Tout cela rend cette approche difficile à utiliser en pratique. Nous avons ensuite vu les techniques de lancer de rayon, simples et intuitives mais non adaptées dans le cas d'un nuage de points. La sélection par volume est une approche prometteuse mais utilise principalement des volumes simples, comme des sphères ou des cubes, rendant la sélection difficilement précise. En revanche utiliser des volumes complètement libres rend la manipulation particulièrement difficile.

L'approche que nous avons retenue est donc d'utiliser une sélection par volume partiellement libre, en contraignant le volume créé de façon à simplifier le plus possible sa création et sa manipulation tout en gardant un niveau de liberté suffisamment élevé pour autoriser une bonne précision.

4.3 Propriétés de conception de l'ESA

4.3.1 Vue d'ensemble

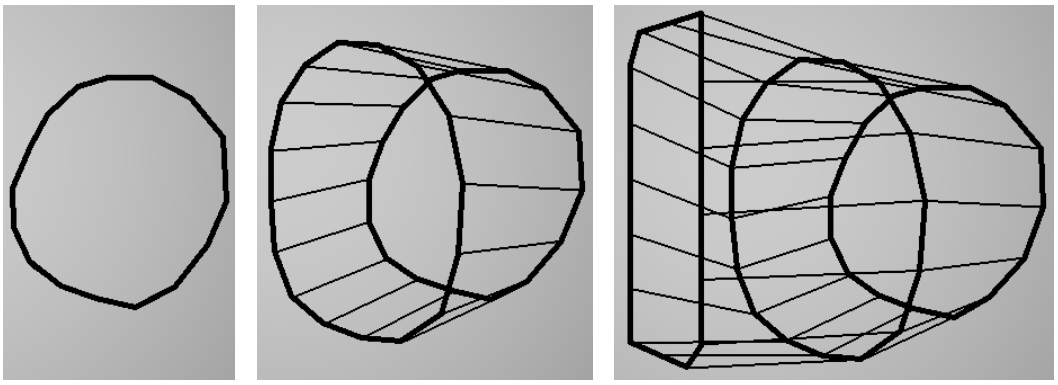
L'ESA est la nouvelle technique de sélection que nous proposons. Elle a été prévue pour être utilisée par tous (experts ou non) sur un ordinateur classique pour sélectionner un ensemble de points dans un nuage de points, correspondant à une forme « complexe ». Nous définissons une forme « complexe » comme une forme difficilement sélectionnable avec les techniques utilisant des volumes existants et standards (une sphère, un cuboïde ou un cône). Il s'agit notamment des cas dans lesquels des volumes irréguliers (comme des zones rouillées) doivent être sélectionnés, des cas où le volume cible correspond à une combinaison de volumes différents (comme le tube d'un télescope à l'intérieur d'un dôme) ou encore des cas où le volume ne correspond pas aux formes géométriques les plus utilisées (comme une pyramide, un trapézoïde...).

L'idée derrière le développement de notre technique, l'ESA, est de permettre une sélection simple et pourtant efficace basée sur une forme ajustable tout en gardant des contraintes du plus faible niveau possible. Pour offrir un compromis entre des formes 3D simples qui ne correspondent pas bien au volume cible et des objets de forme libre trop difficiles à contrôler, notre approche consiste à construire une enveloppe de façon à ce qu'elle corresponde quasiment au volume cible. Dans le domaine médical, l'approche la plus souvent utilisée est de dessiner un ensemble de contours 2D sur une succession de plans parallèles. Notre approche est similaire mais n'est pas limitée à l'utilisation de plans parallèles lors du dessin d'un nouveau contour.

4.3.2 Utilisation de l'ESA

La figure 4.2 montre un exemple de construction de l'ESA. Elle montre une enveloppe composée de trois contours 2D dessinés librement et reliés par des lignes droites pour former un volume. Une fois qu'une enveloppe est définie, tout point se trouvant à l'intérieur est sélectionné.

Le dessin d'un contour est réalisé sur le « plan de dessin » dont la position et l'orientation peuvent être ajustées indépendamment du point de vue de la caméra



(a) Un premier contour (b) Deux contours (c) Un troisième et différent contour ajouté

FIGURE 4.2 – L'enveloppe de l'ESA

(figure 4.7). Nous avons choisi de les séparer à cause du problème de perception de la profondeur (voir section 2.3). Une fois que le plan de dessin est placé à la position voulue, il est possible d'ajuster automatiquement la caméra pour la mettre en face du plan de dessin (figure 4.8). Un contour peut ensuite être dessiné librement sur le plan de dessin autour de la section visible du volume ciblé (figure 4.2a).

Utiliser la technique de l'ESA s'appuie donc sur de multiples itérations d'un processus à deux étapes : 1) placer le plan de dessin à l'endroit désiré dans le nuage de points et 2) dessiner un contour 2D sur ce plan (liant automatiquement le contour courant au précédent). Une fois achevée, l'enveloppe créée doit être validée pour terminer sa sélection.

À travers la première étape du processus, de multiples contours sont créés et joints. Deux contours successifs peuvent très bien être de formes différentes (comme la figure 4.2c le montre).

Une fois que l'enveloppe finale est validée, tous les points à l'intérieur deviennent sélectionnés. Une enveloppe est donc composée d'un minimum de deux contours sans être limité à un nombre maximum.

La forme est similaire en principe au cylindre généralisé de [Vinayak *et al.* 2013]. Elle est cependant dans notre cas définie par une succession de points connectés par des lignes droites au lieu de formules mathématiques complexes à établir.

4.3.3 Implémentation de la sélection de points

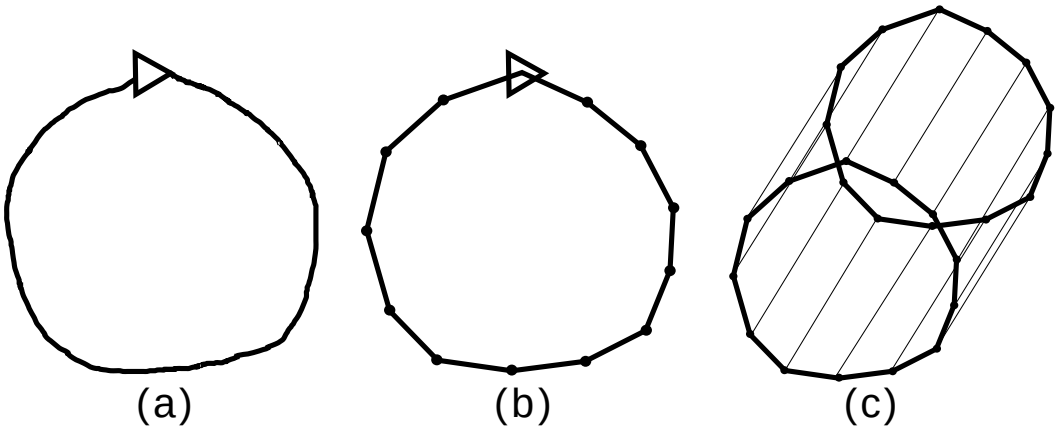


FIGURE 4.3 – Transformation contour vers section de l'ESA

La transformation des multiples contours dessinés en enveloppe se fait en plusieurs étapes (figure 4.3). La première est de convertir un contour (4.3a), qui est un ensemble de pixels convertis en position 3D sur le plan de dessin, en un ensemble de points définissant le contour. Pour ce faire, en premier lieu la longueur du contour est calculée, puis le contour est échantillonné régulièrement par un nombre prédéfini de points (4.3b). Ce nombre est un paramètre de l'application, et dans l'implémentation de test est constant quelque soit le contour (360 points ont été utilisés pour les tests). Du nombre de points dépends alors la précision avec laquelle le contour dessiné est proche du contour de l'ESA créé. Un nombre de points élevé augmentera la fidélité, mais augmentera la charge de calcul (notamment pour déterminer si un point est à l'intérieur). Vient ensuite le problème d'alignement des différents contours. En effet, si l'ensemble des points ne tournent pas dans le même sens (horaire ou anti-horaire) ou bien si l'origine des contours n'est pas la même, la liaison ne sera pas parfaite. Pour cela, la normale du plan utilisé pour le contour courant est comparée à la précédente. Si elle sont différentes, le sens de rotation courant est inversé. Enfin, pour déterminer le point de départ du contour, le point le plus proche en angle d'un vecteur représentant le « haut » et le plus éloigné en hauteur de l'origine (moyenne des points du contour) est choisi. Les lignes sont enfin formées par les points deux à deux de deux contours (4.3c), en commençant par leur point de départ.

Le problème principal dans l'utilisation d'un objet de forme libre est de déterminer si oui ou non un point est à l'intérieur de la forme. Pour cela, nous avons

Algorithme 2 Déterminer si un point est à l'intérieur

```
// triangles est un tableau contenant tous les triangles d'une enveloppe
fonction EST_DEDANS(triangles, point)
    direction ← (0,1,0) // la direction n'est pas importante
    nombreIntersection ← 0

    pour tout triangle dans triangles faire
        si INTERSECTE(triangle, point, direction) alors
            nombreIntersection ← nombreIntersection + 1
        fin si
    fin pour
    si (nombreIntersection modulo 2) = 1 alors
        retourner TRUE
    sinon
        retourner FALSE
    fin si
fin fonction
```

utilisé un algorithme qui fonctionne pour n'importe quel maillage fermé (une surface représentée par des triangles) et donc applicable à l'ESA. Nous envoyons un rayon à partir de chaque point du nuage. Si le rayon rencontre un nombre impair de faces de l'enveloppe, alors le point est à l'intérieur (algorithme 2).

Bien que cet algorithme fonctionne bien, le coût en calcul n'est pas négligeable car il doit être fait pour chaque point. Pour garder notre algorithme en temps réel, le calcul est réalisé sur le GPU en utilisant les *compute shaders* de OpenGL. Le GPU étant extrêmement parallélisé, il y a l'équivalent d'un *thread* par point et le calcul se fait plus rapidement. Étant donné que ce calcul n'arrive que lorsqu'un contour complet est ajouté et que notre maillage, l'enveloppe, contient un nombre relativement faible de faces, il n'a pas été nécessaire de pousser plus loin l'optimisation. Sinon une piste aurait été d'utiliser une structure accélératrice comme un *octree* pour stocker les points de l'enveloppe.

4.4 Application de validation de l'ESA

Afin de pouvoir évaluer ses performances, nous avons intégré l'ESA dans une application de rendu 3D utilisée pour visualiser et naviguer dans un nuage de points. Nous avons également inséré différents éléments d'interface de chaque côté de la fenêtre principale pour aider l'utilisateur.

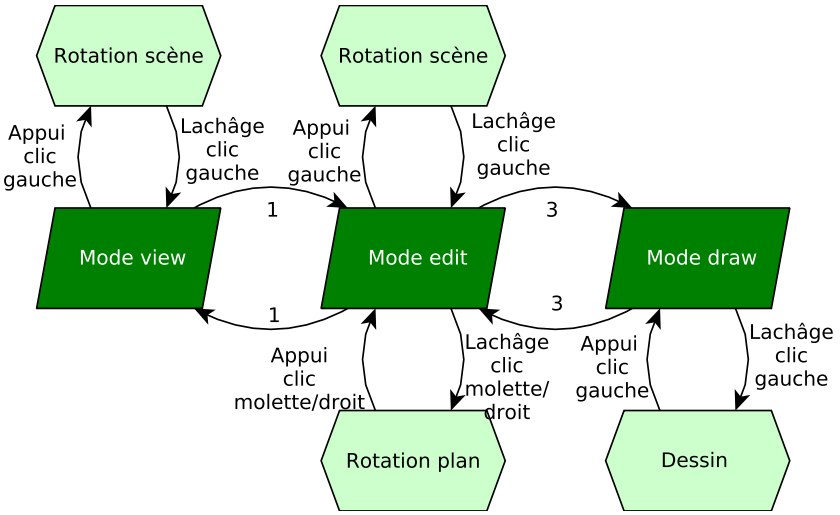


FIGURE 4.4 – Diagramme d'état

Pour créer une enveloppe avec l'ESA, trois modes d'interaction différents sont offerts à un utilisateur. Chacun de ces modes possède ses fonctionnalités spécifiques, comme résumé dans le diagramme d'états (voir figure 4.4). Le mode « view » permet ainsi seulement de tourner la scène, alors que le mode « edit » permet en plus la manipulation du plan de dessin, et enfin le mode « draw » permet de dessiner un contour. L'interface montre le mode courant au centre, tout en fournissant une zone de rappel sur la gauche (où l'utilisateur peut voir les raccourcis clavier courants par exemple) et une pré-visualisation des autres modes sur la droite (figure 4.6).

Nous décrivons ensuite comment l'ESA fonctionne, c'est-à-dire comment l'enveloppe est construite et contrôlée, en présentant les fonctionnalités offertes dans chacun des trois modes d'interaction.

4.4.1 Mode « visu »

Avant de passer dans l'un des trois modes suivants, l'application offre un mode un peu particulier, le mode « visu » (figure 4.5). En effet, celui-ci n'a de sens et de raison d'être que dans le cadre d'une utilisation de l'ESA dans un état expérimental. Il n'est donc pas libre d'accès pour l'utilisateur. Il apparaît uniquement quand une nouvelle forme est affichée, et ne permet que la rotation de la caméra, comme dans le mode « view ». C'est donc le premier mode qu'un utilisateur voit. Une fois que l'utilisateur le désire, l'application passe en mode « view » et ce mode n'est

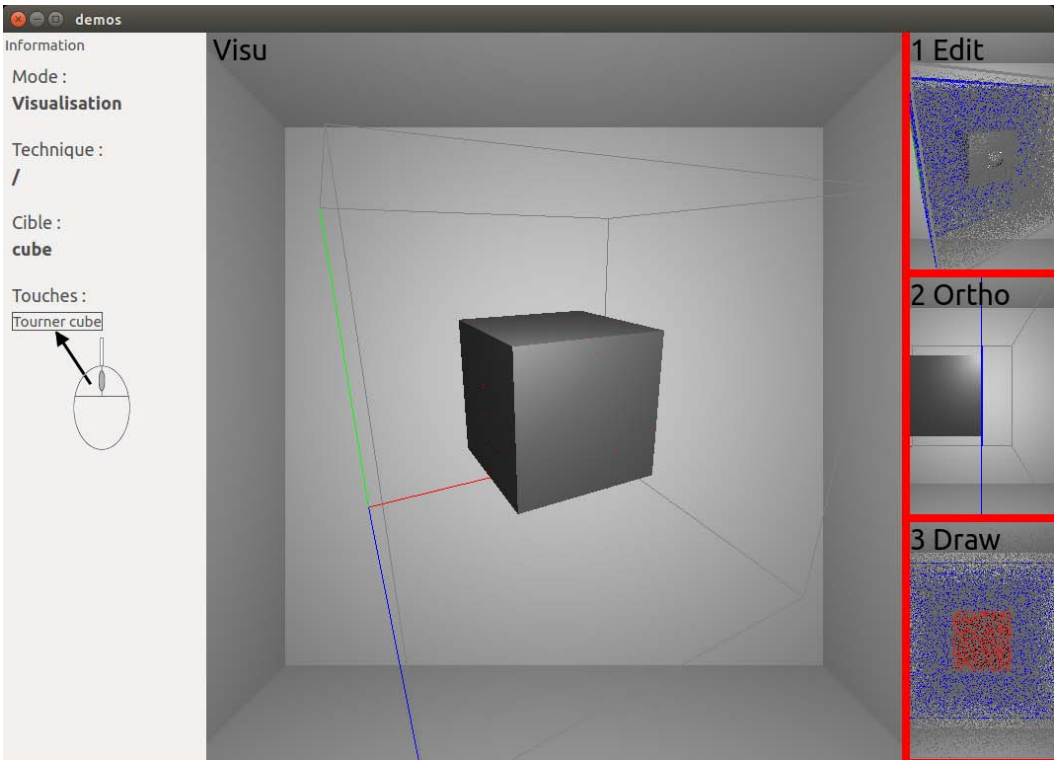


FIGURE 4.5 – Application de test, mode « visu »

plus accessible. Il affiche uniquement une représentation de la surface de l'objet cible, sans la gêne des points qui représentent le bruit. Il n'existe que pour permettre à l'utilisateur de bien se représenter la cible, et de réfléchir à la manière dont la sélection sera effectuée.

4.4.2 Mode « view »

Dans le mode « view » (figure 4.6), la scène 3D et le volume cible sont représentés, ainsi que la sélection courante si elle existe. Le code couleur est rouge pour les points cibles, bleu pour les autres points, verts pour les points cibles sélectionnés et blanc pour les autres points sélectionnés. Le but est donc, une fois la sélection terminée, d'avoir le maximum possible de points verts et bleus, et le minimum de points rouges et blancs.

Ce mode permet la rotation de la scène 3D autour de son centre, ainsi que le transfert vers le mode « edit ». Il permet également la suppression de tout le volume de sélection (recommençant ainsi la sélection à zéro), ou uniquement du

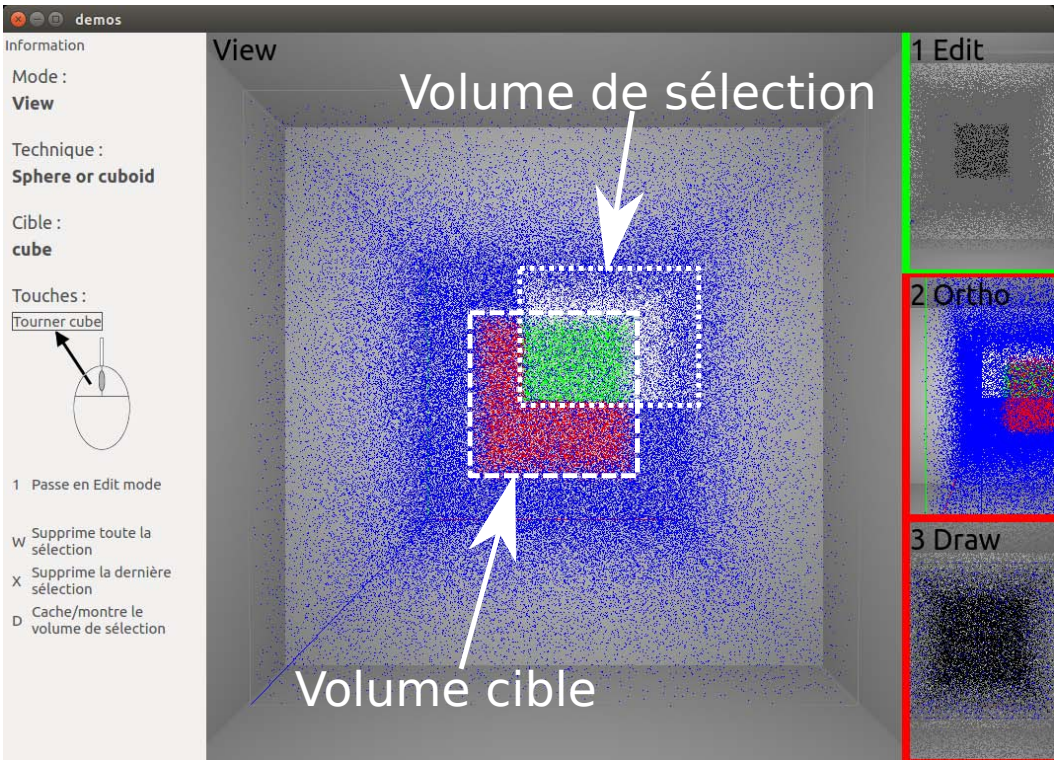


FIGURE 4.6 – Application de test, mode « view »

dernier contour dessiné (action qu'il est possible de répéter successivement tant qu'un contour existe), ainsi que de cacher ou montrer le volume de sélection réalisé qui peut gêner la visualisation de la couleur des points.

4.4.3 Mode « edit »

Le mode « edit » (figure 4.7) fournit la représentation du plan de dessin et en permet la manipulation. Ce mode permet les mêmes manipulations que le mode « view ». Il offre de plus l'interface nécessaire pour contrôler le plan de dessin, sa position (en offrant la possibilité de le déplacer dans le sens de sa normale visible en blanc sur la figure) et sa rotation (selon les deux angles possibles contrôlés indépendamment, le centre de rotation étant représenté par un carré blanc visible de coté sur la figure).

Enfin il permet le passage vers les deux autres modes, « view » et « draw ». D'autres fonctions existent pour aider l'utilisateur à réaliser sa sélection. Il est tout d'abord possible de remettre le plan de dessin à sa position originale, c'est-à-

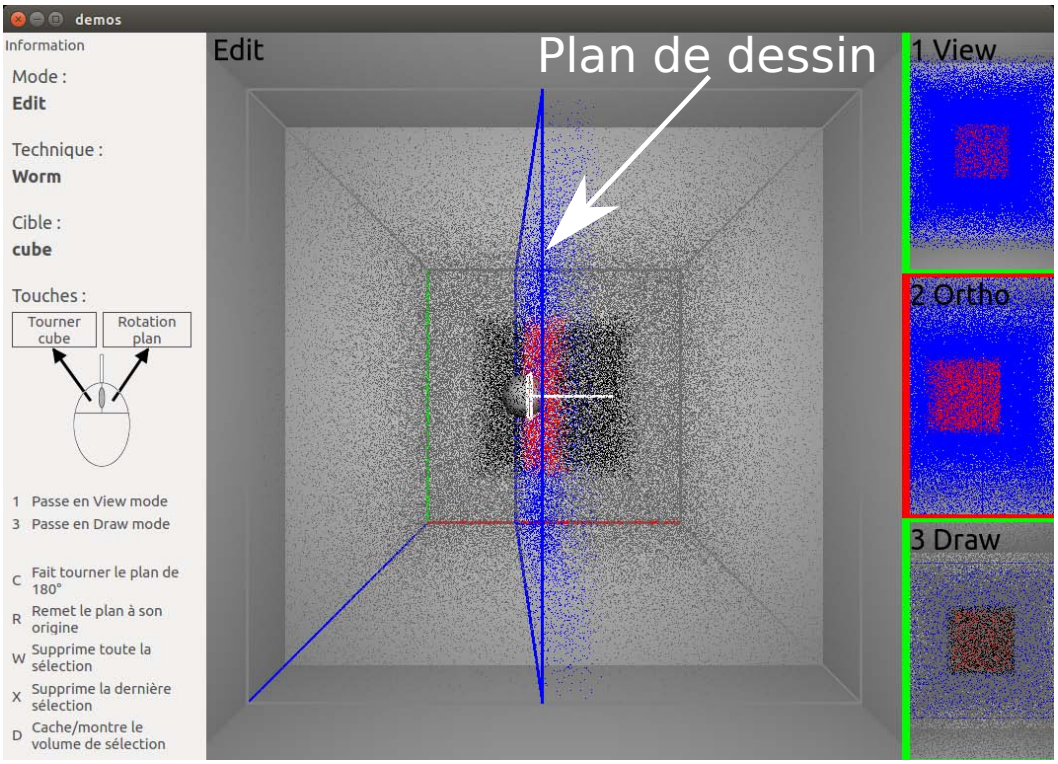


FIGURE 4.7 – Application de test, mode « edit »

dire au bord du volume à sélectionner et avec la rotation par défaut. Il est également possible de changer le côté où la couleur des points est affiché. Cela permet par exemple, lorsque le plan de dessin est au bord du volume à sélectionner, de voir d'un coté le début de ce volume, et de l'autre uniquement le bruit.

Les couleurs des points de ce mode sont les mêmes que le mode « view », mais uniquement d'un coté du plan et sur une épaisseur prédéfinie et fixe. Cela permet de mieux voir les points près du plan, là où le contour sera dessiné. Le reste des points est affiché en noir s'ils font partie de la cible, ou en gris sinon.

La sphère visible proche du centre sur la figure 4.7 fourni un indice à l'utilisateur. Elle représente le centre de la vue du mode « draw ».

4.4.4 Mode « draw »

Le mode « draw » (figure 4.8) permet le dessin du contour. La caméra dans ce mode est toujours placée de façon à voir le plan de dessin, et toutes les parties déjà dessinées du volume de sélection sont cachées de manière à garder la cible

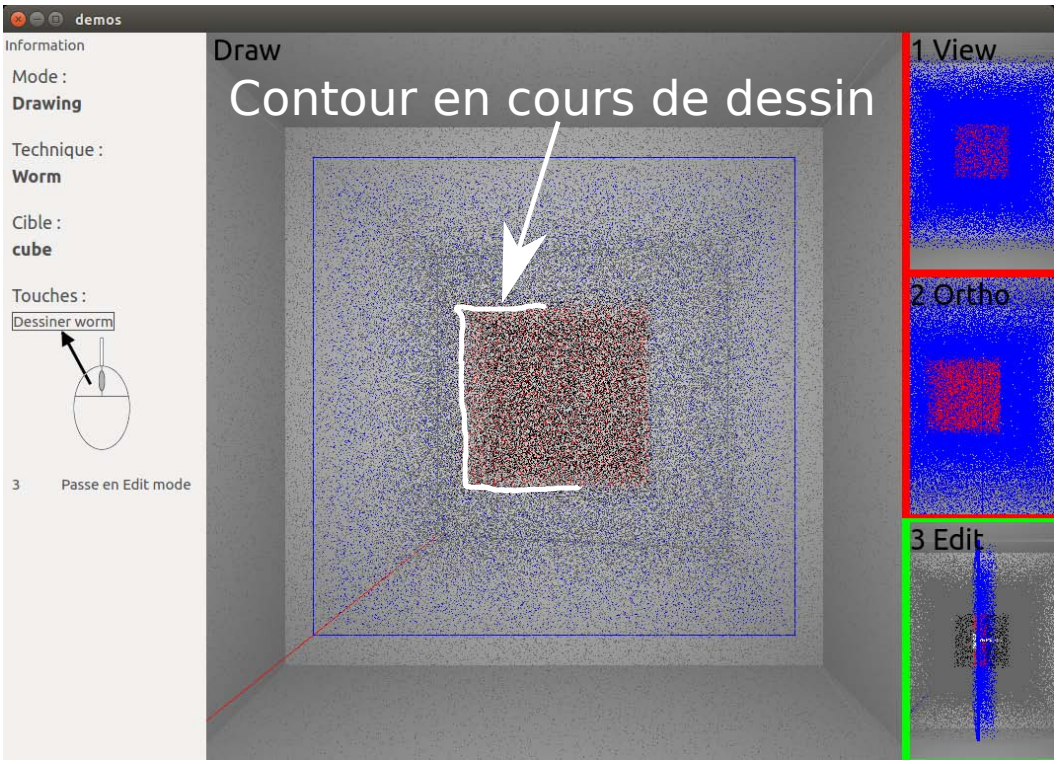


FIGURE 4.8 – Application de test, mode « draw »

visible. Dans ce mode il est uniquement possible de dessiner et valider un nouveau contour, puis de repasser vers le mode « edit ». La couleur des points est la même que pour le mode « edit ».

4.4.5 Vue « ortho »

Enfin, une dernière aide est fournie à l'utilisateur. Il ne s'agit pas d'un mode, mais uniquement d'une vue affichée au centre du bandeau de droite. Elle montre un point de vue toujours orthogonal au plan de dessin, c'est-à-dire que la vue est toujours parfaitement alignée avec la tranche du plan (qui est représenté par le trait rouge au centre de la vue). Cette vue permet de placer le plan précisément à l'endroit désiré. Elle permet également de déplacer le plan précisément, en cliquant dessus tout en déplaçant la souris.

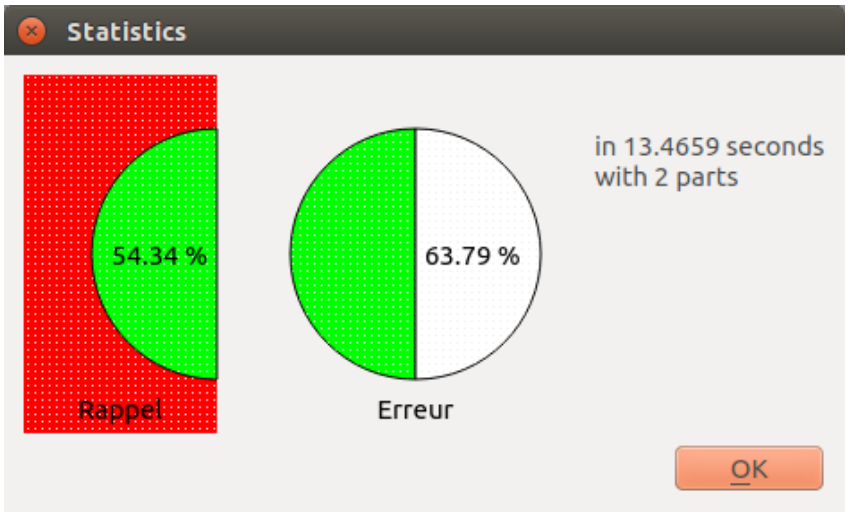


FIGURE 4.9 – Application de test, affichage des résultats

4.4.6 Affichage des résultats

Pour terminer le processus de sélection, et donc valider le volume, la sélection courante doit être déclarée finale. Si l'option est activée, cela entraîne l'affichage des résultats via un popup visible figure 4.9. Pendant l'évaluation expérimentale, cette fenêtre ne s'affichait qu'à la fin d'une sélection d'entraînement. Ce popup affiche plusieurs statistiques :

- le rappel, qui permet de savoir quel pourcentage de points cibles a été sélectionné (les points verts) ; par exemple si tous les points de la cible (correspondant aux points rouges dans la scène 3D) sont sélectionnés, le rappel sera de 100%,
- l'erreur (l'inverse de la précision), qui permet de savoir, parmi les points sélectionnés, le pourcentage de points non cibles (correspondants aux points blancs) ; si autant de points rouges que de points bleus ont été sélectionnés, l'erreur sera de 50%,
- le temps mis pour la sélection (à partir du premier tracé de contour jusqu'au dernier),
- le nombre de contours dessinés.

4.4.7 Interaction avec l'application

Cette section détaille les choix faits pour l'interaction avec l'utilisateur, c'est-à-dire les touches assignées à chaque fonction disponible.

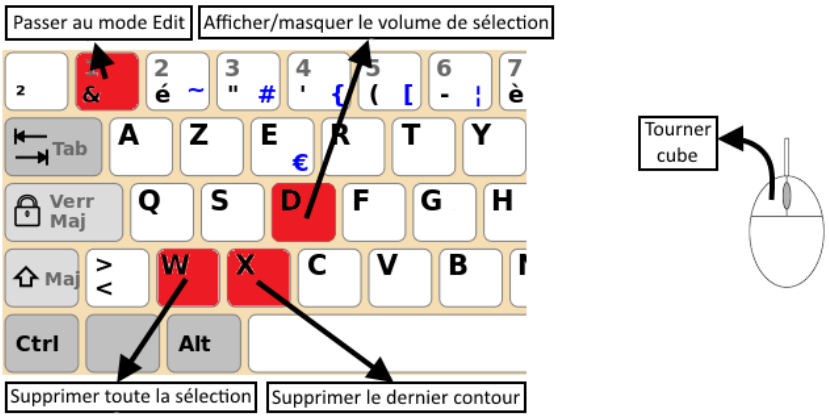


FIGURE 4.10 – Application de test, touches du mode « view »

Pour le mode « view », quatre touches sont disponibles en plus du clic gauche de la souris (voir figure 4.10). En plus de la touche pour changer de mode, il est possible de supprimer tout ou une partie de la sélection réalisée, ainsi que de cacher ou montrer le volume de sélection, ce qui permet de visualiser sans gêne les points à l'intérieur de ce volume.

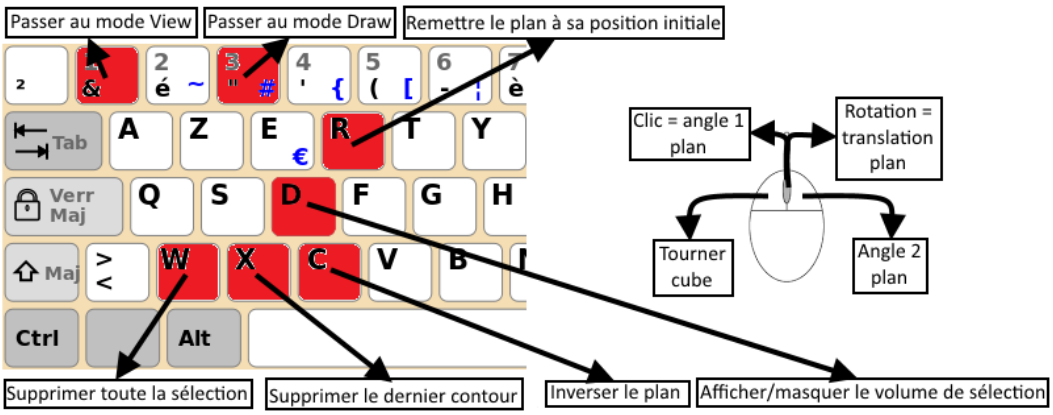


FIGURE 4.11 – Application de test, touches du mode « edit »

Le mode « edit » propose les mêmes touches que le mode « view », mais en rajoute pour le contrôle du plan de dessin. Ainsi il est possible de remettre le plan à ses position et orientation originales, mais aussi de le retourner simplement à

l'appui d'une touche. Pour la souris nous avons ajouté la manipulation du plan, que ce soit sa translation ou ses deux rotations (voir figure 4.11).

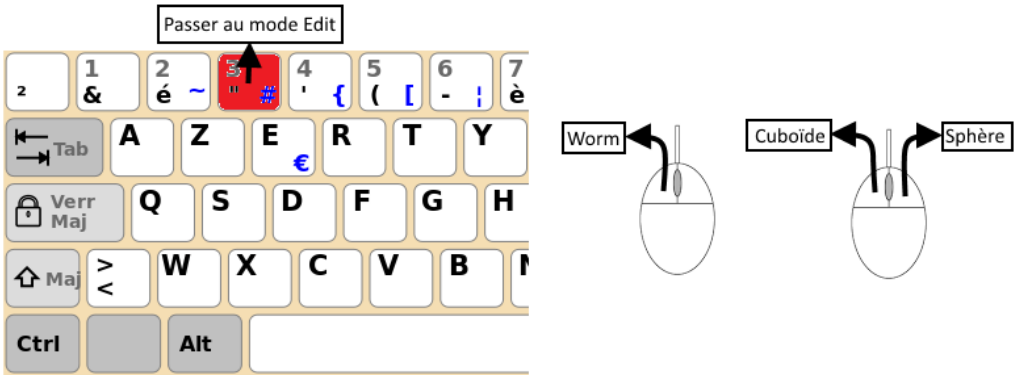


FIGURE 4.12 – Application de test, touches du mode « draw »

Enfin le mode « draw »ne permet que le dessin d'une enveloppe, en fonction de la technique choisie, en plus de changer de mode (voir figure 4.12).

4.5 Évaluation expérimentale

Afin d'évaluer notre technique et de comparer ses performances avec d'autres, nous avons réalisé deux évaluations expérimentales suivant le même protocole. Les techniques d'interaction avancées n'ont pas été utilisées car elles apportent de la fatigue et ne sont pas encore établies comme référence pour la construction de modèles 3D. Des exemples sont les les solutions *mid-air*, comme [Reuter *et al.* 2010], qui permettent l'interaction dans l'environnement virtuel 3D par l'intermédiaire de mouvements dans l'environnement réel, avec ou sans appareils, également en 3D. La première technique comparée à l'ESA est celle utilisée dans le domaine médical, où le volume est construit par une succession de contours toujours parallèles entre eux. La deuxième s'appuie sur l'utilisation de volumes prédéfinis, qui sont régulièrement utilisés dans l'état de l'art. Concrètement, utiliser cette seconde technique consiste à construire le volume de sélection comme un ensemble de volumes en combinant des sphères et des cuboïdes de tailles, positions et orientations paramétrables.

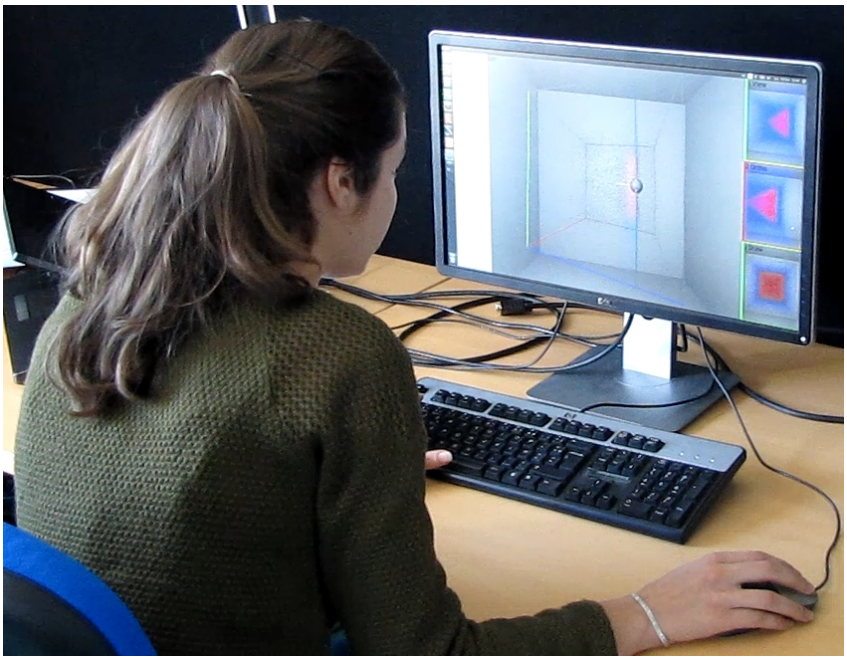


FIGURE 4.13 – Le cadre de l'étude utilisateur

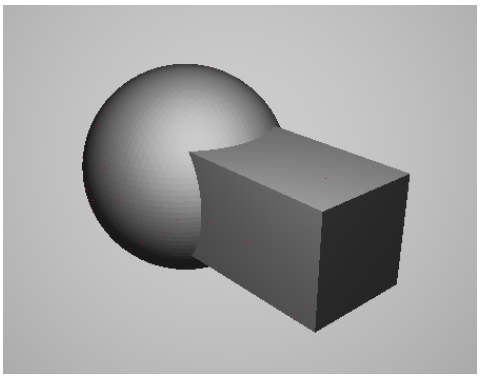
4.5.1 Plate-forme

Les expériences ont été réalisées sur un ordinateur exécutant Ubuntu 14.10 64 bits, avec une carte graphique AMD FirePro M4000. Une sortie image était connectée à un écran 24 pouces (résolution 1920 x 1080). Un clavier et une souris avec 2 boutons et une molette étaient utilisés en entrée (figure 4.13).

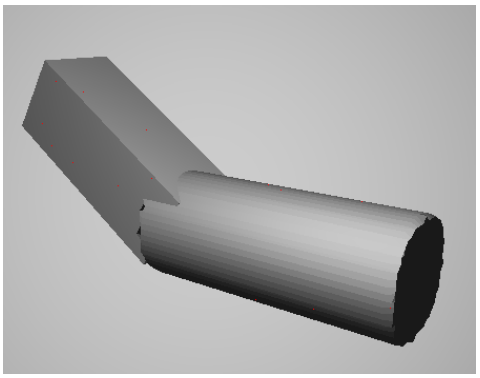
Le nuage de points a été généré procéduralement. Il est constitué de points uniformément répartis dans l'espace, avec un petit décalage aléatoire pour éviter un alignement trop parfait des points qui nuirait à la perception.

4.5.2 Tâche et instructions

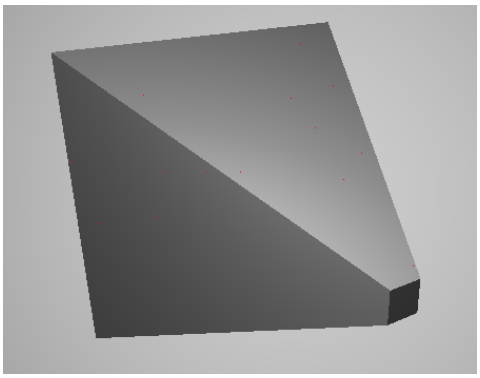
La tâche consistait à sélectionner un ensemble de points prédéfinis (colorés en rouge) dans la scène 3D. Les participants reçurent comme instruction de sélectionner les points cibles le plus précisément et rapidement possible.



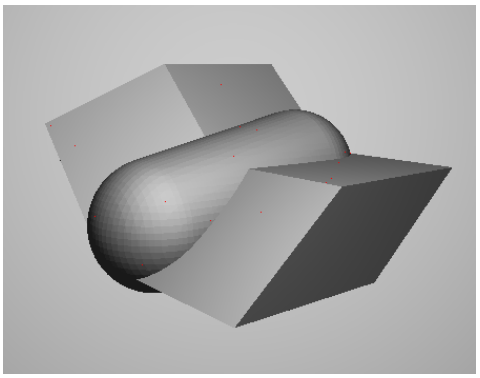
(a) Forme sculpture



(b) Forme L



(c) Forme pyramidale



(d) Forme soudure

FIGURE 4.14 – Les formes à sélectionner

4.5.3 Apparatus

La sélection par volume se passe dans une scène 3D composée d'un simple cube rempli de points et affiché dans un environnement gris (voir figure 4.6). Les volumes cibles sont des formes complexes (voir section 4.3.1) (figures 4.14c et 4.14d), choisies car elles ne correspondent pas à une composition de volumes classiques (sphère, cube...).

Les deux premières formes (visibles en haut de la figure 4.14) ont été utilisées pour l'entraînement. La première, la « sculpture », est simplement composée d'une sphère et d'un cuboïde joints ensemble (figure 4.14a). Elle permettait aux participants d'apprendre à dessiner un contour correspondant à la section changeante du volume cible. La seconde, le « L », est composée d'un cuboïde et d'un cylindre alignés selon différents axes (figure 4.14b). Elle fut conçue pour l'apprentissage de la manipulation du plan de dessin et de ses rotations.

La forme « pyramidale » (S1), utilisée pour l'expérience, est une pyramide avec son sommet coupé, c'est-à-dire ne correspondant pas à un volume classique (figure 4.14c). Enfin l'autre forme utilisée pour l'expérience, la « soudure » (S2), est une combinaison de trois parties : deux trapèzes alignés selon différents axes et connectés par un cylindre arrondi (4.14d), figurant ainsi un élément soudé.

Pour l'expérience seule la rotation de la scène 3D était possible, sans déplacement ni zoom.

4.5.4 Procédure

Une session d'entraînement était réalisée par les participants pour chaque technique de sélection. Elle consistait à sélectionner chaque forme d'entraînement trois fois (toujours dans le même ordre, sculpture puis L). Une fois l'entraînement pour une technique terminé, le participant passait à l'expérience contrôlée visant à la sélection de la forme cible avec la technique courante.

Une représentation de la surface du volume cible était affichée avant le début des trois essais de sélection (comme montré figure 4.14). Cela permettait de percevoir clairement la forme à sélectionner. Le participant était seulement autorisé à orienter la caméra dans la scène pendant aussi longtemps que nécessaire. Dès que le participant le demandait, la sélection démarrait et cette représentation initiale n'était plus accessible.

Chaque fois qu'un participant commençait une sélection, la caméra et le plan de dessin étaient remis dans leur position et orientation par défaut. Chaque fois qu'un participant validait une sélection, aucune cible n'était affichée ensuite. Cela permettait au participant de prendre une pause si besoin. La cible suivante était rendue visible (et le temps compté) à la demande du participant. Il pouvait donc prendre autant de repos que souhaité.

4.5.5 Données collectées

Nous avons mesuré le temps pour réaliser l'entraînement et chaque essai. Nous avons enregistré l'enveloppe créée pour sélectionner le volume de façon à calculer :

— le pourcentage de précision (combien de points sélectionnés sont effecti-

vement des cibles),
— le pourcentage de rappel (combien de points cibles sont effectivement sélectionnés),
— le nombre de formes constituant le volume de sélection créé (correspondant au nombre de contours pour l'ESA).

Nous avons collecté aussi les préférences du participant à travers un classement des deux techniques à la fin de l'expérience, et nous avons demandé, pour les deux techniques, trois points positifs et trois points négatifs. Enfin nous avons réalisé un test SUS [Brooke 1996], qui permet de mesurer l'utilisabilité d'une interface.

4.6 Première évaluation expérimentale

Pour la première évaluation expérimentale, nous avons comparé l'ESA (T1) à une technique que nous avons nommée Enveloppe de Contours Parallèles (ECP) (T2). Elle consiste en une technique existant déjà dans l'état de l'art du domaine médical, qui consiste à créer une enveloppe par succession de contours parallèles.

Dans notre implémentation, l'ECP diffère de l'ESA uniquement dans le mode « edit ». En effet, pour l'ECP, les rotations du plans sont bloquées.

4.6.1 Participants

Douze participants (7 hommes et 5 femmes), âgés de 29.2 ans (écart-type $\sigma = 5.0$) en moyenne ont été recrutés. Tous utilisaient un ordinateur tous les jours, et étaient au minimum familiers avec les environnements 3D, c'est-à-dire capable de comprendre une scène 3D.

4.6.2 Conception

Cette expérimentation suit une conception 2×2 inter-sujet avec les techniques d'interaction et les volumes cibles comme facteurs. Deux techniques de sélection ont été considérées : l'ESA (T1) et l'ECP (T2). Deux formes différentes ont été utilisées comme volumes cibles : la pyramide (S1) et la soudure (S2).

L'expérience est faite de 4 blocs. Chaque bloc comporte successivement deux

paires « technique » et « forme » (T1-S1 / T2-S2, T1-S2 / T2-S1, T2-S1 / T1-S2, T2-S2 / T1-S1). Chaque participant n'est impliqué que dans un bloc. Les participants de différents blocs étaient donc exposés aux deux techniques d'interaction et aux deux formes mais pas dans le même ordre ni avec la même paire « technique » et « forme », pour éviter un effet du à l'apprentissage d'une technique ou d'une forme. Les blocs étaient contre-balancés entre participants et chaque bloc a été fait par trois participants.

Chaque participant avait donc à réaliser un total de 2 techniques d'interaction × 1 forme (par technique) = 2 sélection par volume (sans l'entraînement). En tout pour l'expérience 12 participants × 2 sélections = 24 sélections ont été réalisées (sans compter l'entraînement).

4.6.3 Résultats

Dans les données collectées, aucune différence significative ni tendance n'a été mesurée entre les deux techniques testées pour le rappel, la précision et le temps. En revanche, et malgré les degrés de liberté supplémentaires pouvant accroître la complexité d'utilisation de la technique, l'ESA a été fortement préférée par les utilisateurs (10 sur 12).

Pour cette raison, et aussi pour la plus grande limitation de l'ECP qui ne permet pas la sélection de toutes les formes comme le tore, par exemple, l'ECP n'a pas été retenue dans la suite de l'expérimentation.

4.7 Seconde évaluation expérimentale

Deux techniques de sélection sont utilisées pour la seconde évaluation : l'ESA (T1), décrite avant, et la Sélection par Formes Prédéfinies (SFP) (T3).

La SFP diffère de l'ESA uniquement dans le mode « draw » (voir section 4.4.4). La SFP a été choisie comme technique de référence car la sphère et le cuboïde sont des formes qui sont utilisées couramment dans la littérature, même encore récemment, par exemple dans les travaux de [Benko & Feiner 2007, Cabral *et al.* 2014, Krammes *et al.* 2014, Lucas *et al.* 2005]. De plus ces formes sont utilisées pour réaliser des sélections dans des logiciels disponibles pour tous comme

CloudCompare [clo].

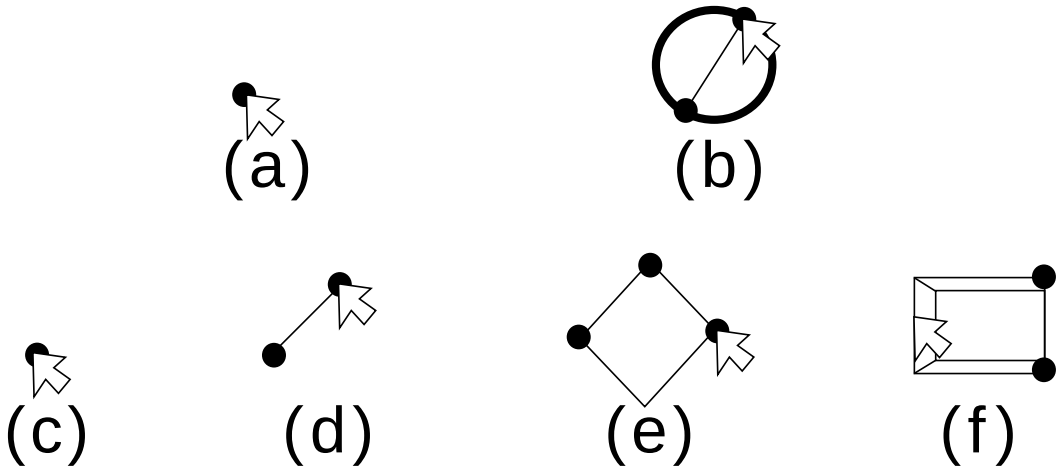


FIGURE 4.15 – Création de sphère ou cuboïde

Dans le mode « draw », il est possible de créer soit une sphère en dessinant deux points représentant son diamètre (figure 4.15, a et b), soit un cuboïde. Sa définition se passe en trois étapes. La première est de définir le côté d'un rectangle de la même manière que pour le diamètre de la sphère (figure 4.15, c et d), puis de choisir la hauteur du rectangle à partir de ce côté (figure 4.15e) pour enfin choisir une profondeur (figure 4.15f), créant ainsi le cuboïde. Pour les deux formes le dessin est interactif dans le sens où un aperçu de la forme est généré en fonction de l'avancement du dessin, aidant ainsi à la précision de la sélection. Pour la dernière étape de création du cuboïde, la caméra se déplace pour s'aligner à l'axe de profondeur du volume en train d'être créé afin de faciliter la définition de sa profondeur (comme le montre la figure 4.15f).

Plusieurs contours (avec l'ESA) ou sphères et cuboïdes (avec la SFP) peuvent ainsi être dessinés pour définir le volume de sélection. Chaque occurrence peut être supprimée comme décrit précédemment pour l'ESA.

4.7.1 Participants

Seize participants (14 hommes et 2 femmes), agés de 30.3 ans ($\sigma = 6.56$) en moyenne ont été recrutés. Tous utilisaient un ordinateur tous les jours, et étaient au minimum familiarisés avec les environnements 3D, c'est-à-dire capable de comprendre une scène 3D.

4.7.2 Conception

Cette expérimentation suit une conception $2 \times 2 \times 3$ inter-sujet avec les techniques d'interaction, les volumes cibles et les répétitions comme facteurs. Deux techniques de sélection ont été considérées : l'ESA (T1) et la SFP (T3). Deux formes différentes ont été utilisées comme volumes cibles : la pyramide (S1) et la soudure (S2).

L'expérience est faite de 4 blocs. Chaque bloc comporte successivement deux paires « technique » et « forme » (T1-S1 / T3-S2, T1-S2 / T3-S1, T3-S1 / T1-S2, T3-S2 / T1-S1). Chaque paire est toujours utilisée trois fois successivement. Chaque participant n'est impliqué que dans un bloc. Les participants de différents blocs étaient donc exposés aux deux techniques d'interaction et aux deux formes mais pas dans le même ordre ni avec la même paire « technique » et « forme ». Les blocs étaient contre-balancés entre participants et chaque bloc a été fait par quatre participants.

Chaque participant avait donc à réaliser un total de 3 répétitions $\times$ 2 techniques d'interaction $\times$ 1 forme (par technique) = 6 sélection par volume (sans l'entraînement). En tout pour l'expérience 16 participants $\times$ 6 sélections = 96 sélections ont été réalisées (sans compter l'entraînement).

Cette expérience suit donc le même protocole que la précédente, excepté que chaque forme est sélectionnée trois fois au lieu d'une unique.

4.7.3 Résultats

Nous présentons dans cette section les résultats quantitatifs et qualitatifs obtenus lors de l'expérimentation.

4.7.3.1 Analyse quantitative

Pour chaque test nous avons vérifié la normalité des données, sur la base de tests de Shapiro-Wilk, pour choisir le test statistique approprié entre paramétré ou non-paramétré.

En premier lieu des tests T ou de Wilcoxon ont confirmé que notre protocole n'a pas entraîné de biais en faveur d'une technique ou d'un groupe. Nous n'avons

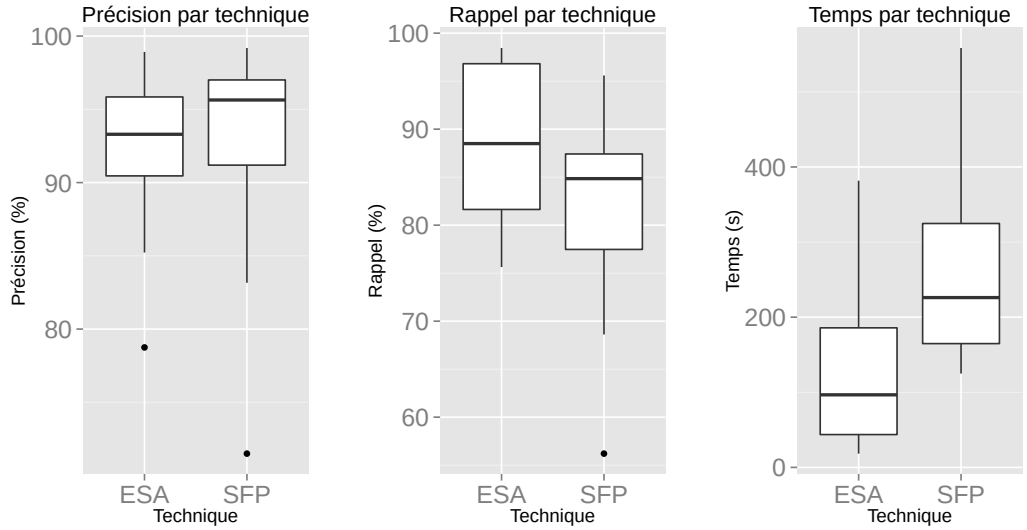


FIGURE 4.16 – Boîte à moustaches des résultats par technique

pas trouvé de différence statistique significative pour les trois mesures utilisées dans cette étude (précision, rappel et temps) pour les groupes :

— commençant par l'ESA (T1) contre la SFP (T3),

— commençant par la pyramide (S1) contre la soudure (S2).

Un test de Kruskal-Wallis confirme aussi qu'il n'y a pas de différence significative entre les quatre groupes. Nous pouvons donc analyser les mesures des quatre groupes comme un seul échantillon.

Critère	Moyenne ESA	Moyenne SFP	p
Temps (s)	133 ($\sigma = 121$)	266 ($\sigma = 154$)	3.9×10^{-2}
Précision (%)	92 ($\sigma = 5.6$)	93 ($\sigma = 9.4$)	non significatif
Rappel (%)	88 ($\sigma = 8.7$)	82 ($\sigma = 11$)	non significatif
Nombre de parties	4.4 ($\sigma = 2.4$)	13.5 ($\sigma = 7.4$)	1.2×10^{-3}

TABLE 4.1 – Résultats par technique

Les résultats de l'expérience sont visibles dans le tableau 4.1. Il montre les résultats obtenus en comparant les deux techniques, indépendamment de la forme. En moyenne nous avons observé que la sélection avec l'ESA était significativement plus rapide qu'avec la SFP (133s contre 266s), ce qui a été confirmé par un test de Wilcoxon ($p = 3.9 \times 10^{-2}$). Il n'y a pas eu de différence significative pour la précision ou le rappel. Ces résultats peuvent également être visualisés avec des boîtes à moustaches sur la figure 4.16. Elles représentent les valeurs extrêmes

(les « moustaches », les lignes et les points), ainsi que les quartiles à 25 et 75% (formant les limites de la boîte) et la médiane (le trait dans la boîte).

Nous avons également observé qu'il y a une différence significative sur le nombre de parties dessinées ($p = 1.2 \times 10^{-3}$; 4.4 sections de l'ESA contre 13.5 sphères ou cuboïdes en moyenne). Enfin il n'y a pas de changement significatif sur le nombre de parties entre les essais pour l'ESA (test de Friedman, $\chi^2(2) = 0.93$, $p = 0.63$) ou pour la SFP (AOV, $F(2, 30) = 0.78$).

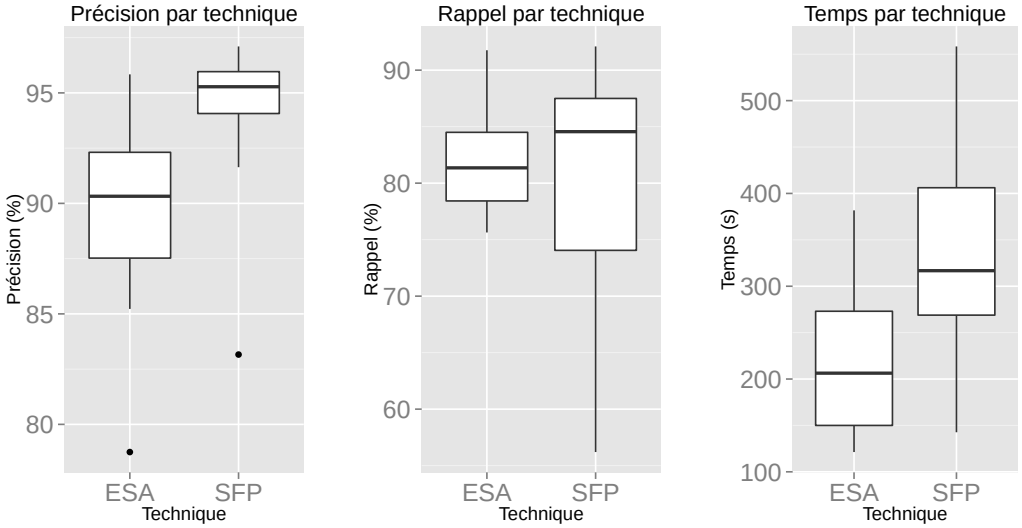


FIGURE 4.17 – Boîte à moustaches des résultats par technique pour la forme « soudure »

Critère	Moyenne ESA	Moyenne SFP	p
Temps (s)	42.9 ($\sigma = 17.2$)	191 ($\sigma = 63.0$)	2.1×10^{-4}
Précision (%)	95 ($\sigma = 2.2$)	92 ($\sigma = 9.3$)	non significatif
Rappel (%)	95 ($\sigma = 5.4$)	84 ($\sigma = 6.5$)	4.7×10^{-3}

TABLE 4.2 – Résultats par technique sur la forme pyramide (S1)

Quand on analyse les résultats en fonction des formes, nous observons que l'ESA permet une bien meilleure sélection de la pyramide (S1) que la SFP, comme montré dans le tableau 4.2. La technique est significativement plus rapide ($p = 2.1 \times 10^{-4}$; 42.9s contre 191s) et avec un meilleur rappel ($p = 4.7 \times 10^{-3}$; 95% contre 84%). Aucune différence significative n'a été observée concernant la précision.

Aucune différence significative n'a été observée entre les deux techniques sur

Critère	Moyenne ESA	Moyenne SFP	p
Temps (s)	224 ($\sigma = 97.9$)	342 ($\sigma = 143$)	non significatif
Précision (%)	89 ($\sigma = 5.3$)	94 ($\sigma = 5.6$)	non significatif
Rappel (%)	82 ($\sigma = 5.6$)	80 ($\sigma = 12$)	non significatif

TABLE 4.3 – Résultats par technique sur la forme soudure (S2)

la sélection de la soudure (S2), comme le montre le tableau 4.3. Il y a seulement une tendance en faveur de l'ESA sur le temps ($p = 7.6 \times 10^{-2}$; 224s contre 342s), et une tendance en faveur de la SFP sur la précision ($p = 6.5 \times 10^{-2}$; 89% contre 94%). Aucune tendance n'est observée concernant le rappel. Les résultats pour la soudure peuvent être visualisés par quartiles sur la figure 4.17.

Critère	Moyenne ESA	Moyenne SFP	p
Temps (s)	162 ($\sigma = 154$)	342 ($\sigma = 168$)	7.6×10^{-3}
Précision (%)	92 ($\sigma = 4.6$)	86 ($\sigma = 9.8$)	non significatif
Rappel (%)	86 ($\sigma = 9.8$)	77 ($\sigma = 11$)	1.1×10^{-2}

TABLE 4.4 – Résultats par technique (uniquement premier essai)

Nous avons également comparé les précédents résultats en ne prenant en compte que le premier essai de chaque technique, indifféremment de la forme, dont les résultats sont donnés par le tableau 4.4. Dans ce cas l'ESA est significativement plus rapide ($p = 7.6 \times 10^{-3}$; 162s contre 342s). Le rappel est également significativement plus élevé pour l'ESA, c'est-à-dire que la technique sélectionne plus de points cibles ($p = 1.1 \times 10^{-2}$; 86% contre 77%). Cependant en termes de précision aucune différence significative n'est observée.

4.7.3.2 Analyse qualitative

Deux aspects ont été considérés pour l'évaluation qualitative : l'utilisabilité et les préférences utilisateur.

Évaluation de l'utilisabilité Le questionnaire SUS [Brooke 1996] donne un score moyen de 74.69 ($\sigma = 17.67$) pour l'ESA. Cette technique d'interaction est donc jugée bonne (« good ») et est considérée acceptable en termes d'utilisabilité [Bangor *et al.* 2008]. La SFP obtient un score de 59.38 ($\sigma = 15.82$). Cette technique d'interaction est jugée « OK » en termes d'utilisabilité et son acceptation est mar-

ginale [Bangor *et al.* 2008]. Un test de Wilcoxon montre que la différence entre les SUS des deux techniques est statistiquement significative ($p = 9.6 \times 10^{-3}$).

Préférences utilisateur À la fin de l'expérience nous avons demandé à chaque participant quelle était leur technique de sélection préférée. Les résultats correspondent avec les scores SUS : l'ESA est largement préférée étant donné qu'un seul parmi les seize participants a choisi la SFP. Enfin, nous avons demandé aux participants de déterminer 3 points positifs et 3 points négatifs à propos des deux techniques.

Les points positifs revenant le plus souvent concernant l'ESA étaient « rapide », « précise » et « facile d'utilisation ». Les points négatifs étaient « imprécise », « nécessite une bonne vision de l'espace » et « inadapté aux sphères, aux vagues et aux formes similaires ». Le dernier commentaire était à propos des objets arrondis. Il est vrai que si l'objet est arrondi dans plus d'une dimension, il requiert alors l'utilisation de plus de sections pour rester précis car chaque section est reliée par des lignes droites. Si c'est seulement dans une dimension, comme pour un cylindre, alors l'ESA n'aura pas de problèmes puisqu'un cercle est relativement facile à dessiner.

Les points positifs revenant le plus souvent concernant la SFP étaient « simple », « rapide à apprendre »et « prévisible ». Les points négatifs cités étaient « pas assez de formes prédéfinies (comme la sphère) », « imprécise » et « besoin de placer précisément le plan de dessin ». Nous remarquons une frustration quand la forme cible ne correspondait pas à un cuboïde ou une sphère, résultant à la demande de plus de formes pour correspondre à chacun des cas. Cependant augmenter ce nombre signifierait aussi augmenter la complexité d'utilisation de la technique.

L'analyse des points positifs et négatifs le plus fréquemment mentionnés révèle que les participants ont fait des commentaires à propos des résultats obtenus lorsqu'ils manipulaient l'ESA, alors que leurs commentaires étaient à propos de la technique lorsqu'ils utilisaient la SFP. Les participants étaient donc plus engagés par la tâche et leur but lorsqu'ils utilisaient l'ESA que la SFP, car ils se concentraient sur le résultat à atteindre et n'étaient pas contraints par des problèmes générés par la technique d'interaction.

4.7.4 Discussion

Parmi les deux techniques d'interaction que nous avons évaluées pour réaliser une tâche de sélection dans un nuage de points 3D, l'ESA est non seulement plus appréciée mais elle offre également de meilleures performances. Une explication possible de ce fait peut résider dans la différence concernant le nombre de parties dessinées. Puisque plus de parties sont nécessaires pour sélectionner une forme avec la SFP, nous pouvons logiquement lui attribuer la différence de temps nécessaire pour réaliser une sélection. Plus de temps et plus de dessins rendent également la technique de sélection plus contraignante, augmentant l'inconfort, ce qui a pour résultat que l'ESA est largement plus appréciée par les participants.

Un autre point est que le nombre de parties dessinées ne varie pas significativement entre les répétitions. Cela signifie qu'une fois qu'un utilisateur a choisi sa manière de sélectionner, il est susceptible de la garder. La première sélection est donc très importante et l'ESA a montré qu'elle était la meilleure sur ce point, sélectionnant plus de points cibles tout en étant plus rapide sans différence significative sur la précision. Elle est aussi très importante car un utilisateur est peu enclin à répéter la même sélection plusieurs fois dans un cas réel.

Pendant l'expérience nous utilisions une projection en perspective pour afficher la scène 3D (comme sur une image prise avec appareil photo). Cela créait des difficultés pour percevoir précisément la position du plan de dessin et donc l'endroit où les participants étaient en train de dessiner exactement. Cela peut être en partie la raison de l'imprécision mentionnée par les participants dans les points négatifs. Une solution à cela serait d'utiliser une projection orthogonale. Elle permet de garder les lignes parallèles visuellement parallèles, mais en perdant tout information concernant la profondeur.

Bien que les résultats établissent que l'ESA est prometteuse, ils montrent aussi que les performances de sélection varient selon la forme cible. La forme « pyramidale » le montre bien ; elle est difficile à sélectionner avec des volumes de sélection classiques (la SFP) mais très simple avec notre technique (l'ESA). De la même manière rien ne sélectionnera mieux une cible sphérique qu'utiliser un volume de sélection sphérique. Avec la forme « soudure » cependant, plus complexe, les différences sont plus floues. L'ESA tend à rester beaucoup plus rapide (35% plus rapide en moyenne), alors que la SFP montre une tendance à obtenir une préci-

sion légèrement meilleure (1% meilleure en moyenne). Aucune tendance n'a été observée concernant le rappel.

De plus amples investigations seront donc nécessaires pour identifier les conditions exactes où l'ESA offre de meilleures performances (temps, précision et/ou rappel). Une définition plus systématique d'une forme complexe sera donc nécessaire.

4.8 Conclusion

Dans ce chapitre nous avons présenté une nouvelle technique d'interaction pour sélectionner des formes complexes dans un nuage de points. Nous avons montré que notre technique, l'ESA, avait des avantages significatifs vis-à-vis d'une technique plus classique, la SFP. Elle offre en moins de temps des niveaux comparables de précision et de rappel et est largement préférée par les participants de notre évaluation expérimentale.

Pour optimiser l'ESA nous envisageons plusieurs possibilités d'améliorations. La première est la possibilité de dessiner plusieurs enveloppes, ce qui n'était pas permis dans notre évaluation. Une seconde évolution consisterait aussi à offrir la possibilité de dupliquer un contour dessiné. Par exemple, pour la sélection d'un cylindre, un utilisateur pourrait d'abord dessiner un cercle puis le copier et le projeter à l'autre bout du cylindre. C'est quelque chose qui n'ajouterait pas beaucoup à la complexité de la technique de sélection, car cela pourrait utiliser des raccourcis bien connus (Ctrl+C, Ctrl+V) mais permettrait probablement d'être encore plus rapide. Enfin, les participants à notre expérience ont exprimé le souhait de pouvoir éditer une enveloppe, c'est-à-dire de permettre de modifier un contour déjà dessiné, ou d'ajouter un nouveau contour entre deux existants. Cela améliorerait probablement la précision et le rappel de la technique.

Un point important est que bien que nous ayons testé notre technique exclusivement sur des nuages de points, elle n'est pas limitée à eux. Puisque notre technique utilise une sélection par volume, toute représentation de données qui est capable de déterminer si une donnée est à l'intérieur ou à l'extérieur d'un volume pourrait être sélectionnée par notre technique. Quelques exemples de telles représentations seraient des polygones, des volumes ou des voxels.

Chapitre 5

Troisième contribution : amélioration par multi-représentation

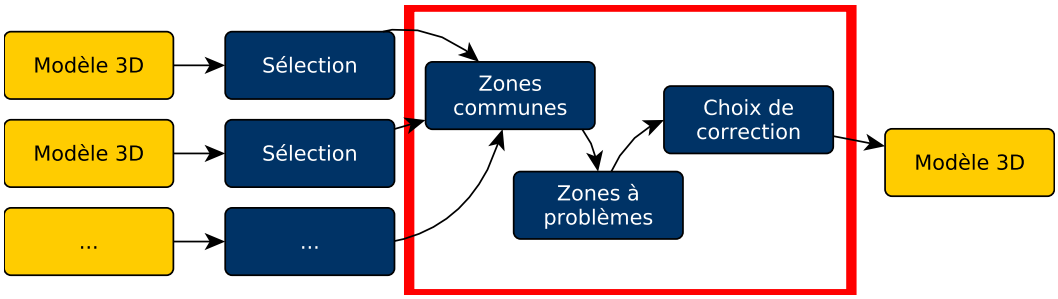


Figure 5.1 – Deuxième partie du processus

Après avoir présenté la première étape du processus dans le chapitre précédent, ce chapitre détaille les trois dernières (entourées en rouge sur la figure 5.1). Il explique en détails la réflexion derrière chaque étape, pourquoi elle est nécessaire et la façon dont nous l'avons mise en œuvre. Il expose ensuite les tests réalisés pour les évaluer et les résultats ayant été obtenus. Il présente enfin des améliorations possibles pour chacune d'elles.

5.1 Vers l'amélioration de nuages de points

Cette thèse ne visant pas à être exhaustive dans le traitement de toutes les représentations possibles et de tous les problèmes possibles, nous avons choisi de nous limiter à un cas exemple, démontrant la faisabilité et l'utilité de notre processus.

Nous nous sommes donc limités à gérer en entrée les nuages de points et les maillages dans le but de générer un nuage de points possédant moins de défauts. De plus, les nuages de points sont des représentations courantes et possédant des défauts dus à leur acquisition. Nous commençons donc par présenter les différents problèmes possibles pour ces représentations et les solutions que nous avons apportées à ces problèmes. Ce chapitre continue avec la présentation de l'application de test et de différents problèmes d'implémentation, et se termine par les tests du processus sur différents modèles et une discussion des résultats.

5.2 Principe

Le principe est de tirer avantages des multi-représentations d'un modèle afin de corriger chacune de celles-ci. Chacune a en effet des avantages et des inconvénients différents, comme cela a été détaillé dans le chapitre 2 État de l'art. Pour cela, nous avons défini trois étapes qui nous permettent de tirer profit des avantages en ignorant les inconvénients, lorsque plusieurs modèles de différentes natures sont disponibles et dans le même repère. Dans le même repère, c'est-à-dire qu'ils possèdent la même taille et orientation et sont placés au même endroit dans l'environnement virtuel. Cela peut être une étape supplémentaire à réaliser avant l'application de notre processus ; elle n'est pas détaillée ici.

5.2.1 Deuxième étape : la détection de zones communes

Afin de pouvoir détecter les zones problématiques il faut en premier lieu détecter les zones où les différents modèles se chevauchent. En effet, bien que représentant le même objet les différents modèles ne sont pas forcément complets, certaines parties pouvant exister dans l'un mais pas dans l'autre des modèles 3D. Cette première partie permet alors de trouver les zones où plusieurs modèles (et représentations) sont disponibles.

La figure 5.2 montre le principe des zones. Si on considère que la figure représente une voiture, nous pouvons voir qu'il y a une partie maillée dans les zones a et b, et une partie couverte par un nuage de points dans les zones b et c. Les zones a et c ne contiennent donc qu'une seule représentation, alors que la zone b en contient deux : c'est une zone commune.

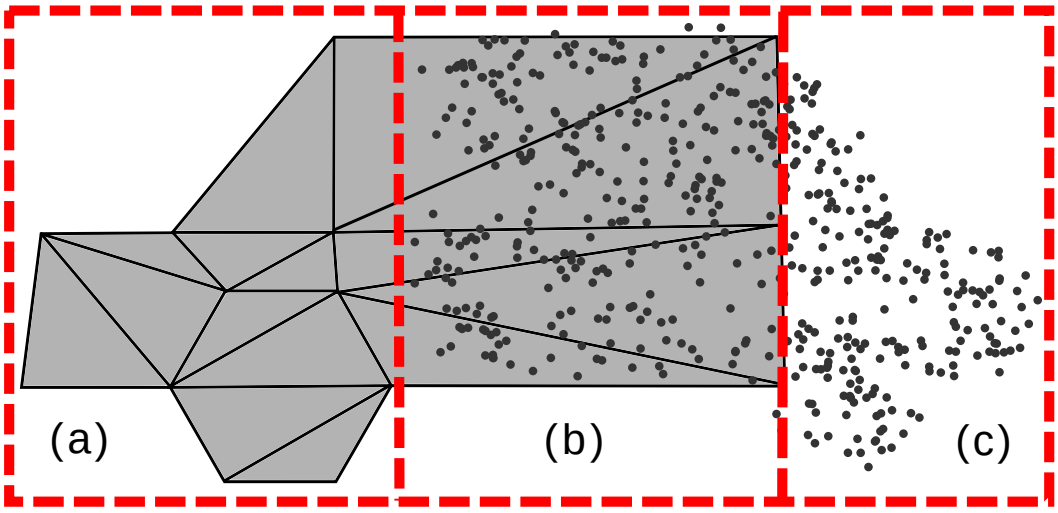


FIGURE 5.2 – Différentes zones (a) 1 modèle (b) Zone commune (c) 1 modèle

5.2.2 Troisième étape : la détection de zones à problèmes

Une fois que les zones communes sont déterminées, l'étape qui suit vise à détecter les zones où des problèmes se posent. Typiquement, si une zone ne contient qu'un seul modèle, c'est une zone possédant un problème parce que tous les modèles ne contiennent pas la même chose, c'est-à-dire que les modèles ne représentent pas exactement le même objet. La nature des différents problèmes potentiels dépend de la nature des modèles disponibles ainsi que celle du modèle qui sera exporté. Nous avons déterminé au moins une partie de ces problèmes dans le cas d'un nuage de points et d'un maillage, comme nous le montrerons plus loin (section 5.3).

5.2.3 Quatrième étape : la correction des zones à problèmes

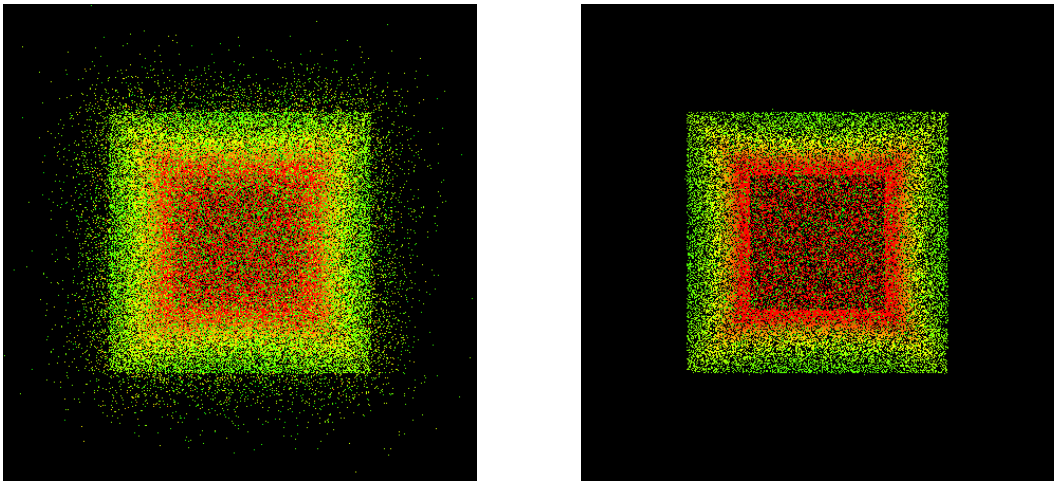
Enfin, une fois que les zones à problèmes sont détectées, il ne reste plus qu'à corriger les défauts rencontrés. Dans cette étape les traitements réalisés vont dépendre des problèmes détectés, et donc encore de la nature des modèles d'entrée et du modèle de sortie. Nous détaillons ainsi les solutions possibles aux problèmes envisagés dans le cadre de nuages de points et de maillages section 5.3.

5.3 Déterminer et traiter les différents problèmes

Une fois le cas d'étude choisi, c'est-à-dire réparer un nuage de points à l'aide de maillages, la première étape a été de déterminer quels sont les problèmes qu'un nuage de points peut présenter. Cette section détaille les problèmes ainsi déterminés, et donne pour chacun un ensemble de solutions permettant de les résoudre.

Les exemples montrés dans cette section sont triviaux et ne visent qu'à expliciter les problèmes et une de leurs solutions. Le maillage utilisé pour ces exemples étant un simple cube, il n'est pas représenté dans les figures. L'application du processus sur des modèles plus complexes est discutée section 5.5 et 5.7.

5.3.1 Premier problème : absence de triangle



(a) Avec bruit (b) Traité

FIGURE 5.3 – Exemple de suppression de bruit

Le premier problème possible dans le cas de représentation en nuage de points et de maillages, ce sont les zones où uniquement le nuage de points est présent. Les points de ces zones n'ont donc pas de triangle correspondant.

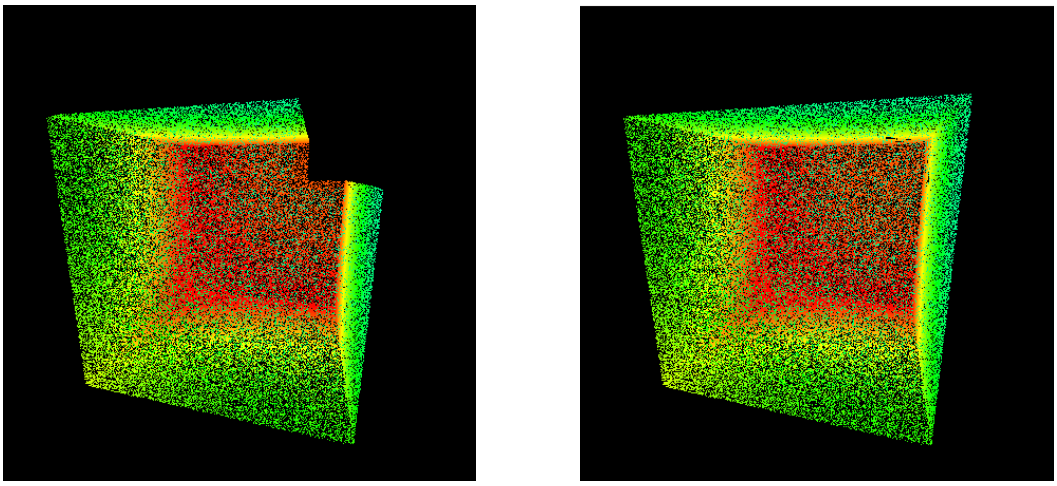
Deux situations possibles ont été déterminées dans ce cas, avec leur solution correspondante :

— l'acquisition par scanner de l'objet représenté est plus complète que le maillage, et les points seront donc gardés comme tels,

— les points concernés ne font pas partie de l'objet représenté, ils sont donc classifiés en tant que bruit et supprimés du modèle final.

La figure 5.3 montre le deuxième cas, où tous les points trop éloignés de la représentation en maillage sont supprimés. La figure sur la gauche (5.3a) montre le nuage de points initial, fortement bruité, et la figure sur la droite (5.3b) montre ce même nuage de points une fois traité pour retirer ce bruit. Les bords sont devenus bien plus nets, à l'extérieur comme à l'intérieur du cube même si ce deuxième aspect est difficilement visible sur la figure.

5.3.2 Deuxième problème : absence de point



(a) Avec un trou (b) Traité

FIGURE 5.4 – Exemple de remplissage

Le deuxième problème possible représente le cas inverse du précédent : les zones où uniquement des triangles sont présents, sans points. Dans ce cas, il existe également deux situations possibles à considérer :

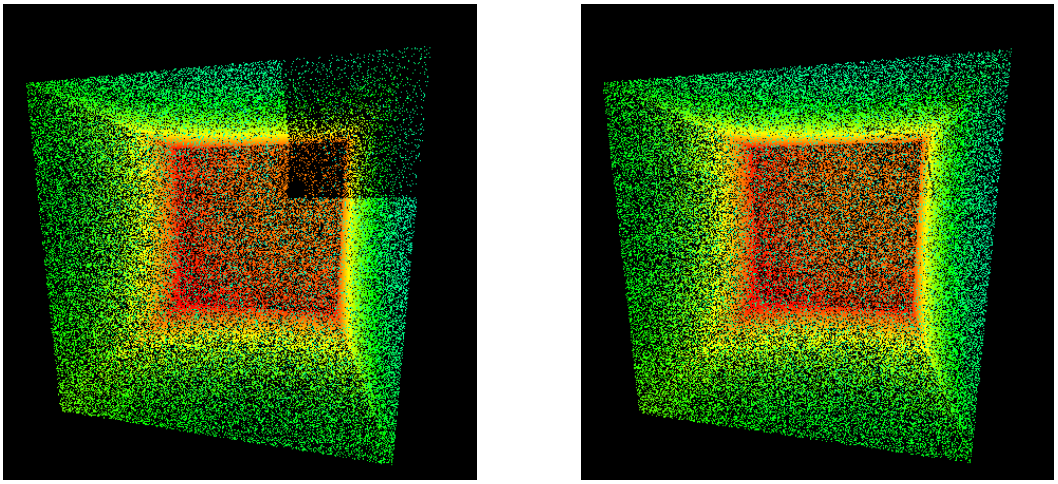
— le nuage de points devrait couvrir ces zones mais ne le fait pas, par exemple à cause d'un problème d'occlusion. La zone est donc échantillonnée pour la remplir de points,

— le modèle ayant évolué dans le temps une partie de l'objet a disparu, la zone est donc gardée telle qu'elle est et les triangles sont ignorés.

La figure 5.4 montre le premier cas. Le modèle de base, visible figure 5.4a, comporte une zone manquante en haut à droite, qui est remplie grâce au maillage par échantillonnage de ses triangles. Le résultat est observable sur la figure 5.4b,

où le trou n'est plus visible, sans qu'il ne soit possible de déterminer où était la frontière entre les deux anciennes zones.

5.3.3 Troisième problème : variation de densité



(a) Avec densité faible (b) Traité

FIGURE 5.5 – Exemple de complétion

Enfin le dernier type de problème concernant les nuages de points est leur possible densité variable. La densité dans notre cas est calculée comme le nombre de points dans une zone donnée, ce qui est mis en facteur avec la surface du modèle maillé dans cette zone (voir algorithme 4). Certaines zones peuvent contenir un grand nombre de points, alors que d'autres très peu, indépendamment du niveau de détail de l'objet représenté dans ces zones. Cette densité varie généralement en fonction de la proximité au scanner qui a permis d'acquérir le nuage de points, les zones proches de celui-ci étant plus densément peuplées de points.

Trop faible densité Cette variation de densité peut se traduire de deux manières, dont la première est le cas où des zones sont de densité plus faible que dans le reste du modèle. Deux situations existent encore dans ce cas de figure :

— le modèle résultat devrait avoir une densité uniforme, la zone est donc complétée en ajoutant des points jusqu'à obtenir la densité appropriée,

— la densité n'est pas un problème et la zone est donc gardée telle qu'elle. La possibilité est offerte même si il existe peu de chance qu'elle soit utile.

La figure 5.5 montre la première situation. La figure de gauche (5.5a) montre que le nuage de points possède une zone de faible densité en haut à droite, et la figure de droite (5.5b) montre le résultat une fois ce modèle complété pour s'ajuster à la densité voulue. Encore une fois, il n'est pas possible de remarquer le traitement effectué sur le modèle amélioré.

Densité trop élevée La deuxième manière dont ce problème s'exprime est par l'existence de zones à densité trop élevée. Dans ce cas les deux situations sont identiques aux précédentes, excepté que le traitement appliqué pour corriger le problème est de simplifier la zone (retirer des points) jusqu'à atteindre la densité appropriée.

5.4 Environnement de mise en œuvre du processus

Afin de pouvoir tester le processus présenté, une application a été développée. Elle implémente ce processus et est capable de détecter et de corriger tous les problèmes mentionnés ci-avant, avec toutes les solutions également mentionnées précédemment. Le comportement par défaut de l'application correspond toujours à la première des deux situations décrites. Toutefois, quelque soit le choix par défaut, il est toujours possible de le changer avant d'appliquer la solution choisie. La densité idéale de l'objet résultat par défaut est la densité moyenne du nuage de points.

Dans cette section, la première partie montre l'interface de l'application de test et la deuxième détaille le fonctionnement interne de celle-ci, montrant comment sont réalisés les traitements nécessaires.

5.4.1 Interface

La figure 5.6 montre une capture d'écran de l'application et de ses différents menus. Le bandeau principal, la barre horizontale visible en haut de la figure, est celui qui gère les transitions entre chaque étape du processus. Il permet donc de passer d'une étape du processus à une autre, et affiche les options relatives à celle-ci. C'est via ce bandeau que le modèle résultat est exporté par exemple. Il permet également de sauvegarder le projet dans l'état courant (et de le rechar-

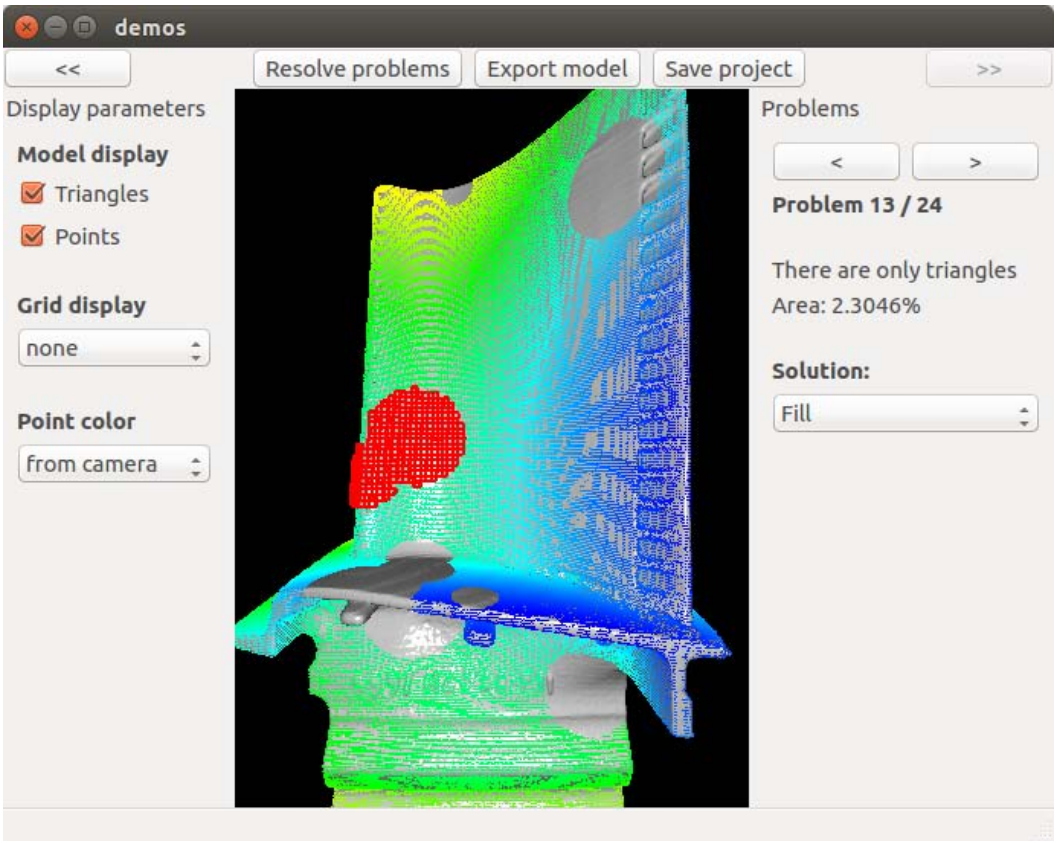


FIGURE 5.6 – Capture d'écran de l'application

ger), évitant ainsi de refaire les calculs de détection de zones communes, par exemple, car ils peuvent être très longs.

La barre de gauche contrôle l'affichage de la partie centrale (montrant le modèle multi-représentations en 3D sur fond noir). Elle permet, de haut en bas, d'abord de montrer ou de cacher les triangles des maillages ou les points des nuages. Elle permet ensuite d'afficher ou de cacher la grille régulière (voir section suivante) en choisissant quelles sont les cases représentées (toutes, uniquement celles avec des points...). Enfin, elle permet de choisir la représentation de la couleur des points du nuage. Il est possible de choisir parmi deux options. La première représente la distance des points à la caméra utilisée pour visualiser les modèles. Les couleurs des points varient du bleu (le plus proche) en passant par des nuances de vert et en finissant par du rouge (le plus éloigné). La deuxième représentation utilise le même code couleur (bleu - vert - rouge) mais représente la distance de chaque point à son triangle le plus proche. Elle représente donc la distance du nuage de points par rapport au maillage. Sur la figure 5.6, c'est la

première représentation qui est utilisée.

La barre de droite, visible uniquement lorsque les zones problématiques ont été détectées, permet de visualiser chaque zone et d'afficher le problème correspondant. Elle permet également de savoir quelle proportion du modèle multi-représentations est concernée par le problème courant, et de changer la solution qui sera appliquée pour cette zone. Sur la figure 5.6, une de ces zones est visible en rouge sur la partie centrale. Le problème de cette zone est qu'elle contient uniquement des triangles.

L'application comporte aussi une barre d'état, en bas de la figure, qui affiche du texte pour renseigner sur l'état courant de l'application et les traitements en train d'être effectués.

Enfin la partie centrale représente les modèles 3D en cours de traitement. Elle permet de naviguer à l'intérieur où à l'extérieur de ceux-ci à l'aide du clavier et de la souris, afin de pouvoir visualiser comme souhaité les modèles et les représentations additionnelles comme la grille régulière ou les zones à problèmes.

5.4.2 Implémentation

Après avoir vu le principe général du processus et les problèmes possibles d'un nuage de points, cette partie détaille comment les traitements nécessaires sont réalisés par l'application de test.

5.4.2.1 Détection de zones communes

La deuxième étape du processus est de détecter les zones communes entre les multi-représentations. Pour cela, il a été décidé d'utiliser une grille régulière, c'est-à-dire un ensemble de cases cubiques de même taille, existant partout où existent des points ou des triangles. Un exemple d'une telle grille est montré figure 5.7, où il est bien visible que la grille n'existe que si un élément à englober existe.

Les différentes zones à déterminer dépendent donc des cellules de la grille. Les zones communes sont chacune des cellules où plusieurs représentations sont présentes.

L'ensemble des traitements dépend de cette grille, un paramètre important est

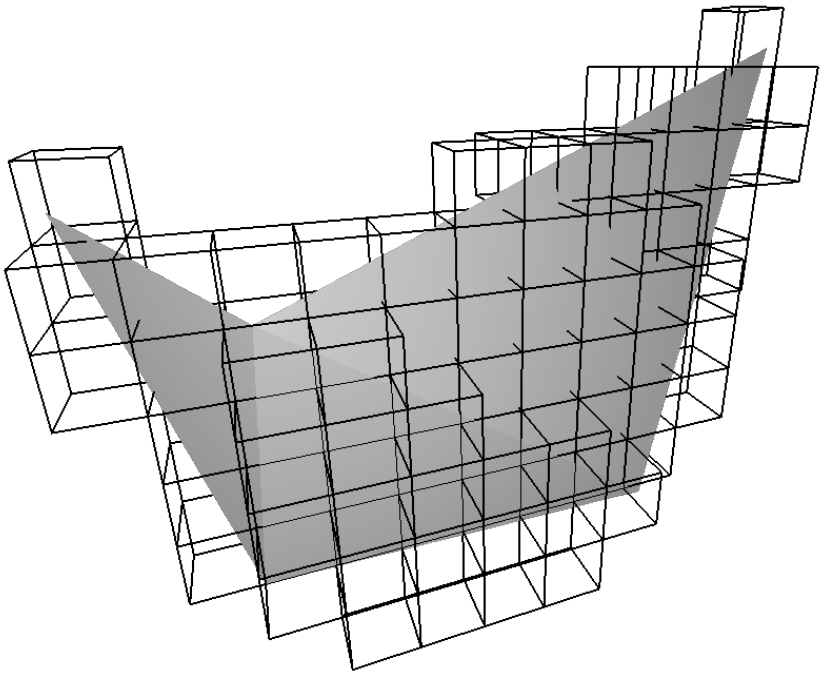


FIGURE 5.7 – Un exemple de grille

donc la taille de ses cellules. En effet, si celle-ci est trop grande la détection ne sera pas précise et de nombreux détails seront perdus, mais si celle-ci est trop faible les zones communes détectées n'auront pas de sens. Il est donc crucial de choisir la bonne taille pour correctement détecter les problèmes. L'application propose une taille par défaut selon la taille globale de la grille, mais permet de choisir la valeur voulue. Un exemple visible ci-après (section 5.6) montre l'effet de la taille de grille sur la détection et donc la résolution des problèmes.

La construction de cette grille est assez simple, si un point ou un triangle existe à la position possible d'une cellule, cette cellule est créée et contient ce point ou ce triangle. Cependant, afin de simplifier les traitements suivants, chaque triangle n'appartenant pas à au plus une cellule est découpé selon la grille régulière. Le résultat est qu'aucun triangle n'appartient à plusieurs cellules, ce qui dans le cas de l'échantillonnage, par exemple, aide car l'ajout de points est limité par cellule, évitant donc un test supplémentaire, en échange d'un temps de construction de la grille plus long.

L'algorithme 3 montre comment ce traitement est réalisé. Le principe est de trouver toutes les cellules contenant une partie d'un triangle. Ensuite, pour cha-

Algorithme 3 Créer la grille et découper les triangles

```
// retourne l'indice de la case de la grille contenant point
fonction POINT_VERS_CASE(point, origine, taille)
    // point et origine sont des points 3D, taille est un flottant
    retourner (point − origine)/taille
fin fonction

// origine est le point zéro de la grille, taille est le côté d'une cellule
fonction AJOUTER_TRIANGLES(triangles, origine, taille)
    grille // la grille régulière
    pour tout triangle dans triangles faire
        boiteEnglobante ← la boite englobante de triangle alignée selon les
axes
        debut ← POINT_VERS_CASE(boiteEnglobante.minimum)
        fin ← POINT_VERS_CASE(boiteEnglobante.maximum)

        // si le triangle n'occupe qu'une seule case
        si debut = fin alors
            AJOUTER(grille[debut.x][debut.y][debut.z], triangle)
        sinon
            pour i entre debut.x et fin.x faire
                pour j entre debut.y et fin.y faire
                    pour k entre debut.z et fin.z faire
                        cellule ← grille[i][j][k]
                        si triangle intersecte avec cellule alors
                            polygone ← INTERSECTION(cellule, triangle)
                            nouveauTriangles ← TRIANGULE(polygone)
                            pour tout t dans nouveauTriangles faire
                                AJOUTER(cellule, t)
                            fin pour
                        fin si
                    fin pour
                fin pour
            fin pour
        fin si
    fin pour
    retourner grille
fin fonction
```

cune de ces cellules, l'intersection avec le triangle est déterminée, donnant un polygone. L'application ne traitant que des triangles, ce polygone est triangulé avant de rajouter chacun des triangles le composant dans sa cellule correspondante. Cet algorithme est ensuite répété pour tous les triangles du modèle en entrée du processus.

5.4.2.2 Détection de zones à problèmes

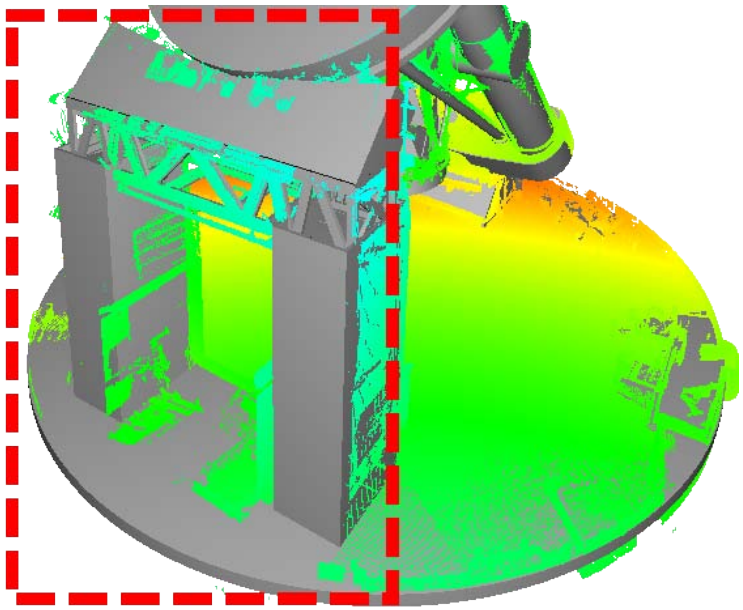


FIGURE 5.8 – Un exemple de zone à problème

Une fois les zones communes trouvées, la détection de zones comportant des problèmes dépend également de la grille régulière. Une zone problématique est définie comme l'ensemble des cellules voisines possédant le même problème, une cellule voisine étant définie comme ayant une face commune (pas de diagonales).

La détection de zones sans point ou sans triangle est assez simple, il suffit de tester leur absence ou leur présence dans chaque cellule. Pour déterminer les zones à forte ou faible densité par contre, il faut d'abord calculer la densité moyenne du modèle multi-représentations. Cette densité dépend du nombre de points mais aussi de la surface totale des triangles, afin d'être plus précis. Il est normal qu'une cellule contienne moins de points si le modèle 3D de l'objet n'a qu'une petite partie présente dans celle-ci.

Algorithme 4 Calcul de densité pour une cellule

```
fonction DENSITÉ(cellule)
    aire ← 0
    pour tout triangle dans cellule.triangles faire
        aire ← aire + AIRE(triangle)
    fin pour
    retourner cellule.points.nombre() ÷ aire
fin fonction
```

L'algorithme 4 montre comment la densité est calculée pour chaque cellule. L'algorithme utilisant les points et les triangles pour ce calcul, il n'est appliqué qu'aux cellules contenant les deux. Le calcul de la densité globale se fait de la même manière, mais en prenant tous les points et tous les triangles des zones communes au lieu de le faire cellule par cellule.

La détection de zones à trop forte ou trop faible densité se fait donc en comparant le densité locale à la densité moyenne. Si la différence est trop grande, cette limite étant un paramètre de l'application, la zone est alors classée comme problématique. Après plusieurs tests, nous avons trouvé qu'une différence de 40% de points en plus ou en moins constituait une bonne limite. Cette valeur est donc celle utilisée par défaut, mais elle reste paramétrable.

5.4.2.3 Enlever le bruit

Une zone constituée uniquement de points est considérée comme problématique ; les points pourraient représenter du bruit. Cependant, se baser uniquement sur les cellules de la grille régulière pour classifier les points en tant que bruit ou non manque de précision. Pour combler ce défaut, l'application offre un autre traitement pour enlever le bruit. Elle propose de calculer la distance minimum entre chaque point et le maillage, en trouvant le triangle le plus proche. Une fois cette distance calculée, il est possible de retirer tous les points plus éloignés qu'une distance donnée en paramètre. Si le maillage représente tout le modèle voulu, c'est une méthode simple et efficace pour retirer le bruit.

5.4.2.4 Échantillonner les triangles

Si le problème d'une zone est qu'elle possède uniquement des triangles et que la solution retenue est de générer des points, le traitement à appliquer est l'échantillonnage des triangles. Un exemple de résultat obtenu par échantillonnage est donné figure 5.9. L'image de gauche (5.9a) montre un simple ensemble de triangles, et l'image de droite (5.9b) montre le résultat de l'échantillonnage tel que réalisé par l'algorithme de l'application de test (algorithme 5).

L'algorithme 5 montre comment ce traitement est réalisé. Son principe est simple : choisir un triangle aléatoirement, puis choisir un point sur ce triangle

(a) Un ensemble de triangles (b) Un ensemble de points

FIGURE 5.9 – Exemple d'échantillonnage

aléatoirement, et enfin répéter jusqu'à obtenir le nombre de points voulu. Cependant, afin d'équilibrer la répartition des points de façon à avoir une densité globale identique, les probabilités de choix ne sont pas égales. En effet, elles dépendent de la surface de chaque triangle ; plus celle-ci est grande, plus le triangle a une probabilité élevée d'être choisi. Cette façon de procéder donne le résultat voulu en utilisant un générateur uniforme de nombres aléatoires (chaque nombre à la même probabilité d'être choisi). Elle permet donc de garder une densité globalement uniforme.

Un autre problème existe cependant, celui de choisir le nombre de points à ajouter. Pour cela, la densité moyenne est utilisée. Il suffit de la multiplier par l'aire des triangles dans la cellule courante pour obtenir le nombre voulu. Ainsi le nombre de points ajoutés reste cohérent par rapport à la présence du modèle dans la cellule et à sa densité moyenne globale.

5.4.2.5 Compléter les zones à faible densité

Compléter les zones se fait exactement de la même manière que l'échantillonnage décrit dans la section précédente. La différence est que le nombre de points à rajouter est calculé de manière à n'ajouter que les points qui manquent. La probabilité des triangles de la zone est également modifiée comme ceci :

$$proba \leftarrow proba \times (1 - densitéCellule \div densitéMoyenne)$$

Ainsi, la probabilité est réduite pour n'ajouter dans la zone que le nombre de points nécessaire pour ajuster la densité défaillante.

Algorithme 5 Échantillonnage

```
// Note : OBTENIR_POINT(triangle, indice) retourne le point du triangle indiqué par indice
fonction ÉCHANTILLONNE(triangle)
    u ← valeur aléatoire entre 0 et 1
    v ← valeur aléatoire entre 0 et 1
    si u + v > 1 alors
        u ← 1 – u
        v ← 1 – v
    fin si
    retourner OBTENIR_POINT(triangle, 0) × u +
            OBTENIR_POINT(triangle, 1) × v +
            OBTENIR_POINT(triangle, 2) × (1 – u – v)
fin fonction

// seuls les triangles à échantillonner sont donnés
fonction ÉCHANTILLONNAGE(triangles, nombreDePoints)
    proba ← 0
    points // tableau
    probabilités // tableau
    // obtient les probabilités d'être choisi pour chacun des triangles
    pour tout triangle dans triangles faire
        proba ← proba + AIRE(triangle)
        ajouter proba dans probabilités
    fin pour
    // génère les points
    pour i = 0 à nombreDePoints faire
        valeur ← valeur aléatoire entre 0 et proba
        triangle ← triangle dans triangles avec la plus petite probabilité au dessus de valeur dans probabilités
        ajouter ÉCHANTILLONNE(triangle) dans points
    fin pour
    retourner points
fin fonction
```

5.4.2.6 Simplifier les zones à haute densité

Si la solution choisie est d'enlever les points en trop, alors la zone concernée est simplifiée. L'algorithme 6, assez simple, montre comment l'application de test retire les points pour obtenir la densité désirée. Elle se contente de choisir aléatoirement un point et de le supprimer, puis de recommencer jusqu'à ce que suffisamment de points aient été retirés. C'est une méthode simple que nous retrouvons dans l'état de l'art (voir section 2.4.1.2).

Algorithme 6 Simplification

```
fonction SIMPLIFIER(zone, nombreDePoints)
    points // tableau résultat
    pour tout cellule dans zone faire
        nombre ← nombre pour cellule dans nombreDePoints
        // transfère nombre points de la cellule choisis aléatoirement, les points
restant étant supprimés
        boucle nombre fois
            i ← nombre aléatoire entre 0 et cellule.nombreDePoints – 1
            AJOUTER(points, cellule.points[i])
            RETIRER(cellule.points, i)
        fin boucle
    fin pour
    retourner points
fin fonction
```

5.5 Résultats sur un cas d'étude : la pale de turbine

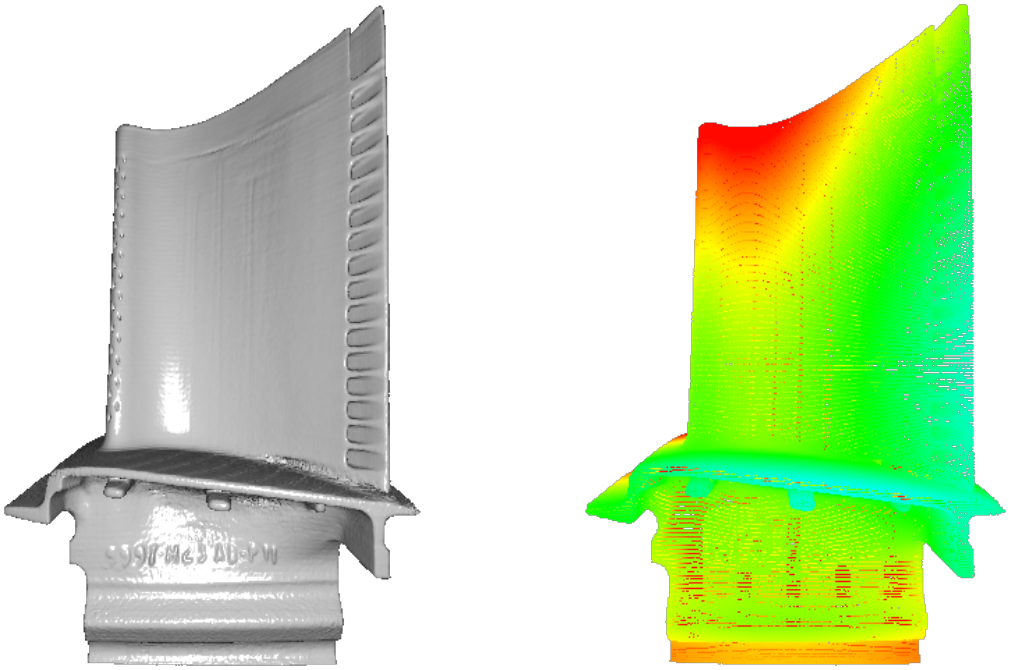
	Original	Trous corrigés	Densité corrigée
Nombre de points (milliers)	883	900	866

TABLE 5.1 – Nombre de points par modèle

Modèle	Avec trous	À densité variable
Nombre de points (milliers)	760	883
Faible densité (%)	10	33
Haute densité (%)	16	24
Trous (%)	16	8

TABLE 5.2 – Analyse des problèmes des modèles

Afin de vérifier qu'à la fois le processus et l'implémentation faite de celui-ci fonctionnent bien, un test a été réalisé sur un modèle connu et déjà de bonne qualité. Cela a permis de comparer les résultats de l'application de test avec ce modèle de bonne qualité. Pour cela, ce modèle a été artificiellement dégradé afin de reproduire les problèmes mentionnés en 5.3. Ce modèle est visible figure 5.10, avec le maillage visible sur la gauche (5.10a) et le nuage de points visible sur la droite (5.10b - les couleurs représentent la distance à la caméra). Il représente une pale de turbine, le nuage contenant environ 883 000 points. Un résumé des

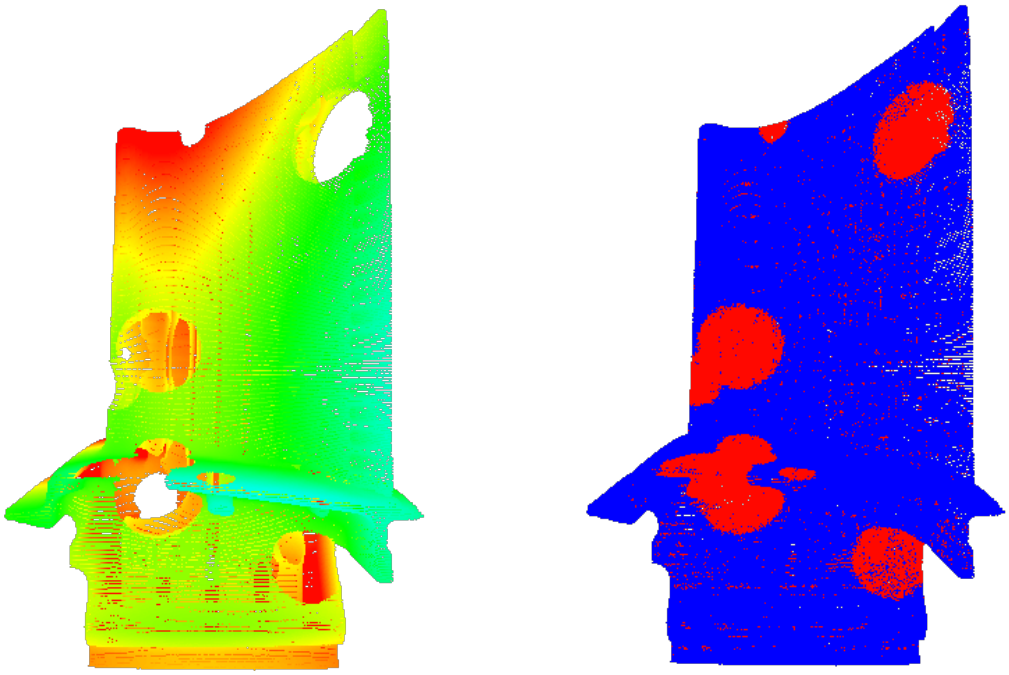


(a) Le maillage (b) Le nuage de points

FIGURE 5.10 – Modèle de pale de turbine issu du *Georgia Institute of Technology's Large Geometric Model Archive* [of Technology]

différents modèles testés et de leur analyse par l'application de test est visible dans les tables 5.1 et 5.2, les résultats étant discutés avec plus de détails par la suite.

Les pourcentages discutés ci-après sont définis en fonction des cellules de la grille régulière, et ne représentent donc pas exactement le modèle (une cellule représente le même pourcentage du modèle multi-représentations qu'elle contienne un point ou mille). Le nuage de points représente parfaitement le modèle maillé et inversement, c'est-à-dire que la distance de chaque point par rapport à son triangle le plus proche est nulle. Nous ne fournissons donc pas de comparaison entre le maillage et le nuage de points. Pour la même raison, nous ne traitons pas le cas de l'enlèvement de bruit, car il serait trivial.



(a) Avec des trous (b) Corrigé

FIGURE 5.11 – Résultat sur le modèle avec trous et après correction

5.5.1 Remplissage des trous

Pour générer le modèle de test troué, nous avons en premier lieu choisi aléatoirement un point du nuage (visible figure 5.10b). Nous avons ensuite, encore aléatoirement, choisi une distance et avons enlevé tous les points présents à l'intérieur de la sphère ainsi créée. Nous avons ensuite répété cette opération vingt fois, et obtenu le modèle visible figure 5.11a contenant environ 760 000 points. Le modèle corrigé par notre processus est visible figure 5.11b, les points bleus représentants les points originaux du modèle avec des trous, et les points rouges montrant les points ajoutés par notre application de test pour corriger les problèmes détectés.

L'application a détecté qu'environ 16% du modèle multi-représentations avait comme problème l'absence de points. Elle a également détecté qu'environ 10% de celui-ci correspondait à des zones de faible densité, ce qui peut être interprété comme les cellules de la grille faisant la transition entre les trous et les zones sans problème. Au total l'application a ajouté environ 140 000 points pour cor-

riger les problèmes. Le modèle corrigé possède donc environ 900 000 points, ce qui, comparé aux 886 000 points du modèle original tend à prouver que le calcul de densité réalisé par l'application pour déterminer le nombre de points à ajouter est correct, les deux chiffres étant relativement proches. De plus, la figure 5.11b montre bien que les points sont rajoutés aux bons endroits, ce qui donne un modèle visuellement plaisant, proche de l'original.

La rugosité est un bon facteur permettant de comparer si deux modèles se ressemblent. Elle permet de déterminer les zones les plus reconnaissables d'un modèle, et donc aide à indiquer si un humain pourra interpréter le modèle traité comme fidèle à l'original. La rugosité du modèle original ainsi que celle du modèle avec trous après correction sont visibles, respectivement, figures 5.15b et 5.15d.

Leur similitude montre que les trous ont été correctement remplis et qu'un humain ne verra pas la différence entre les deux modèles. De plus, comme l'indiquait le nombre de points proche des deux modèles, il n'y a pas de différence sensible entre la densité du modèle original (figure 5.14b) et celle du modèle traité (5.14d). L'application a donc bien respecté les contraintes à la fois du nuage de points et du maillage, pour fournir un modèle corrigé satisfaisant.

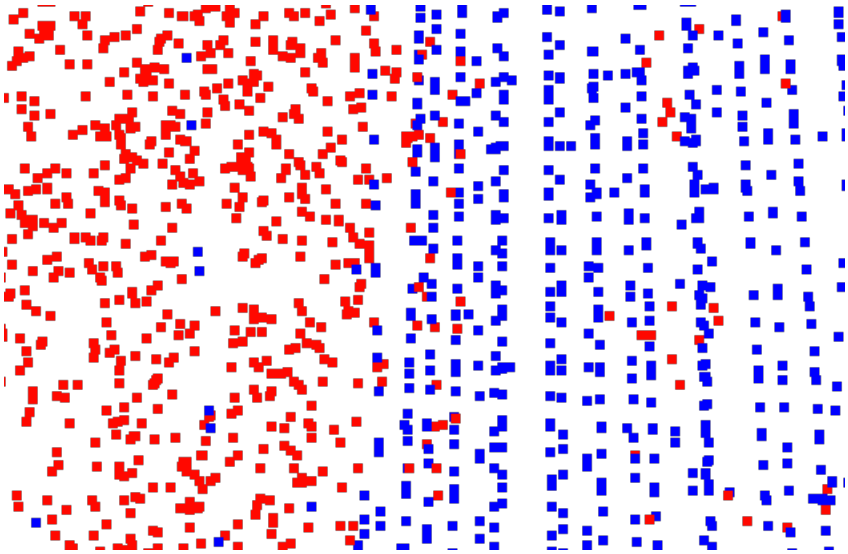
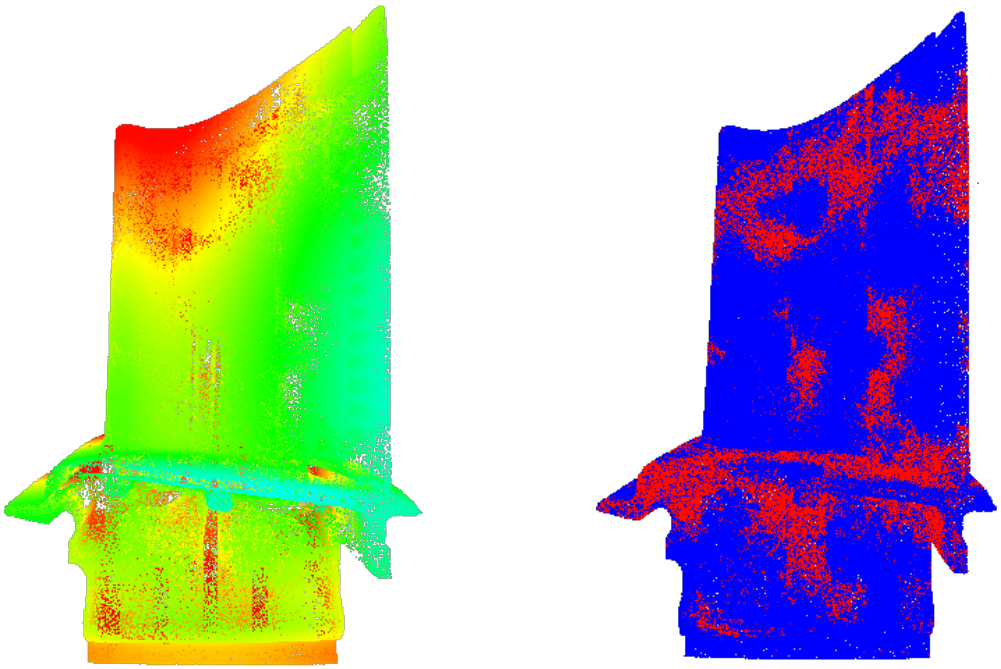


FIGURE 5.12 – Zoom sur une frontière de trou

Cependant le modèle utilisé en exemple est particulier. En effet, tous les points le composant sont parfaitement alignés entre eux. L'application ajoutant les points aléatoirement, aucun motif particulier n'est discernable. La différence est mon-

trée figure 5.12, où les points rouges (ceux ajoutés) n'ont pas de motif et les points bleus (originaux) sont parfaitement alignés. Le résultat est que les zones remplies sont reconnaissables après le traitement, la subtile différence permettant de discerner la frontière entre les deux zones. Cette différence n'est pas visible si aucun motif n'existe, comme le montre le modèle du TBL provenant d'un vrai scanner section 5.7.

5.5.2 Correction de la densité



(a) Avec une densité variable (b) Corrigé

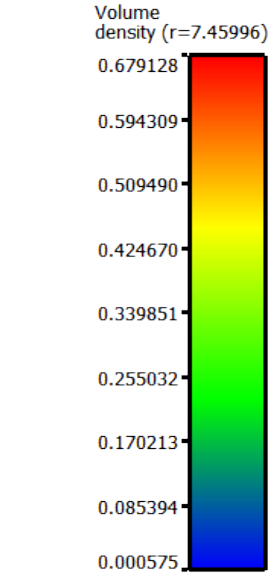
FIGURE 5.13 – Résultat sur le modèle avec une densité variable

Pour générer le modèle possédant une densité variable, nous avons pris pour base le maillage et l'avons échantillonné de la même manière que décrit en 5.4.2.4 pour remplir les trous. Cependant, afin de faire varier la densité, les probabilités de chaque triangle étaient modifiées en fonction de sa position dans l'espace. Pour déterminer cette modification de probabilité, nous avons englobé le modèle dans une grille régulière avec de larges cellules (pour obtenir une différence visible dans la densité). Cette grille a ensuite été remplie de valeurs aléatoires, en favorisant les valeurs extrêmes (pour obtenir une densité beaucoup plus faible ou beau-

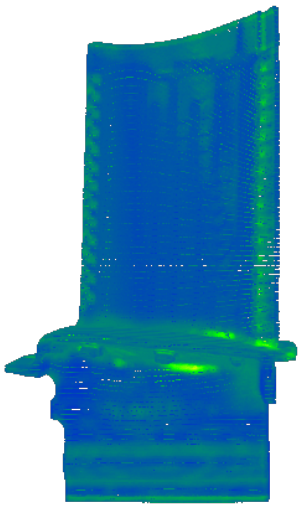
coup plus forte). Ensuite, pour chaque triangle était calculée sa position moyenne, utilisant cette valeur pour lire dans la grille la modification de probabilité. Afin d'éviter des « cubes » de densité visibles, la valeur récupérée était moyennée par les cellules voisines, en fonction de la distance du point à celles-ci. L'échantillonnage a ensuite été réalisé avec le même nombre de points que le nuage original.

Le modèle généré pour ce problème est visible figure 5.13a, sa correction par l'application étant montré figure 5.13b (avec en rouge les points ajoutés et en bleu les points conservés). L'application de test a détecté qu'environ 33% du modèle multi-représentations avait une densité faible, alors que 24% était classé en densité élevée. Pour corriger cela, environ 229 000 points ont été ajoutés et 246 000 ont été enlevés. Encore une fois nous pouvons constater que le calcul de la densité semble correct, puisque le nuage résultat comporte quasiment le même nombre de points que l'original.

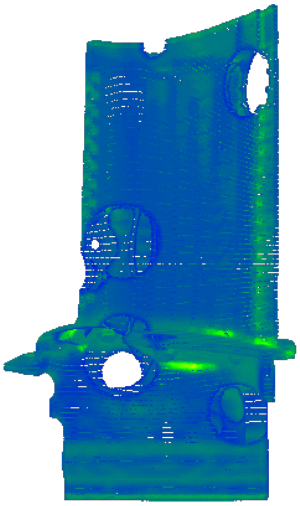
Nous pouvons, de plus, constater, que la densité ne semble pas affecter la rugosité, comme le montre les figures 5.15e (le modèle altéré) et 5.15f (le modèle traité), toutes les deux proches du modèle original (5.15b). La variation de densité n'affecte donc pas la capacité d'un humain à reconnaître l'objet représenté. Elle présente toutefois des artefacts peu plaisants visuellement, comme visible figure 5.13a, en plus de ne pas être idéale pour différents traitements comme la reconstruction. La figure 5.14e montre le calcul de densité sur le modèle artificiellement altéré. Elle permet de se rendre compte que le modèle est loin d'être parfait. Ce même calcul sur le modèle traité (figure 5.14f) montre que l'application de test a su faire disparaître les différences pour obtenir un modèle avec beaucoup moins de défauts. Il est en plus également très proche du modèle original (figure 5.14b). L'application de test a donc une nouvelle fois prouvé sa capacité à corriger les problèmes d'un nuage de points.



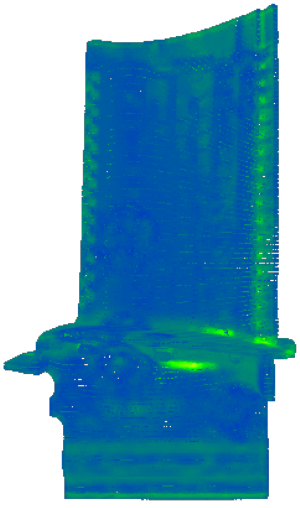
(a) Échelle



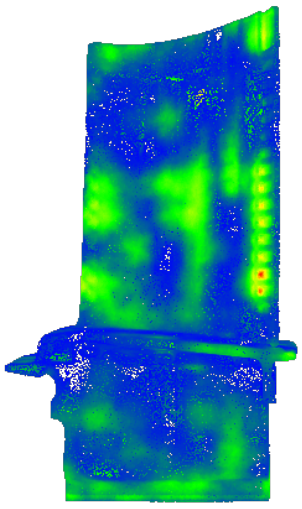
(b) Original



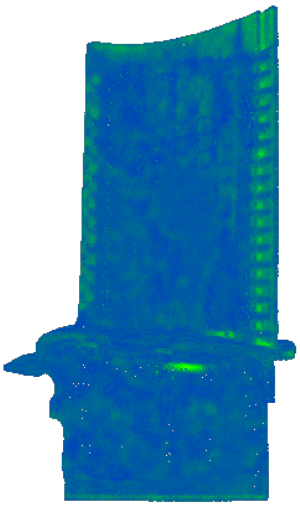
(c) Avec trous



(d) Trous corrigés

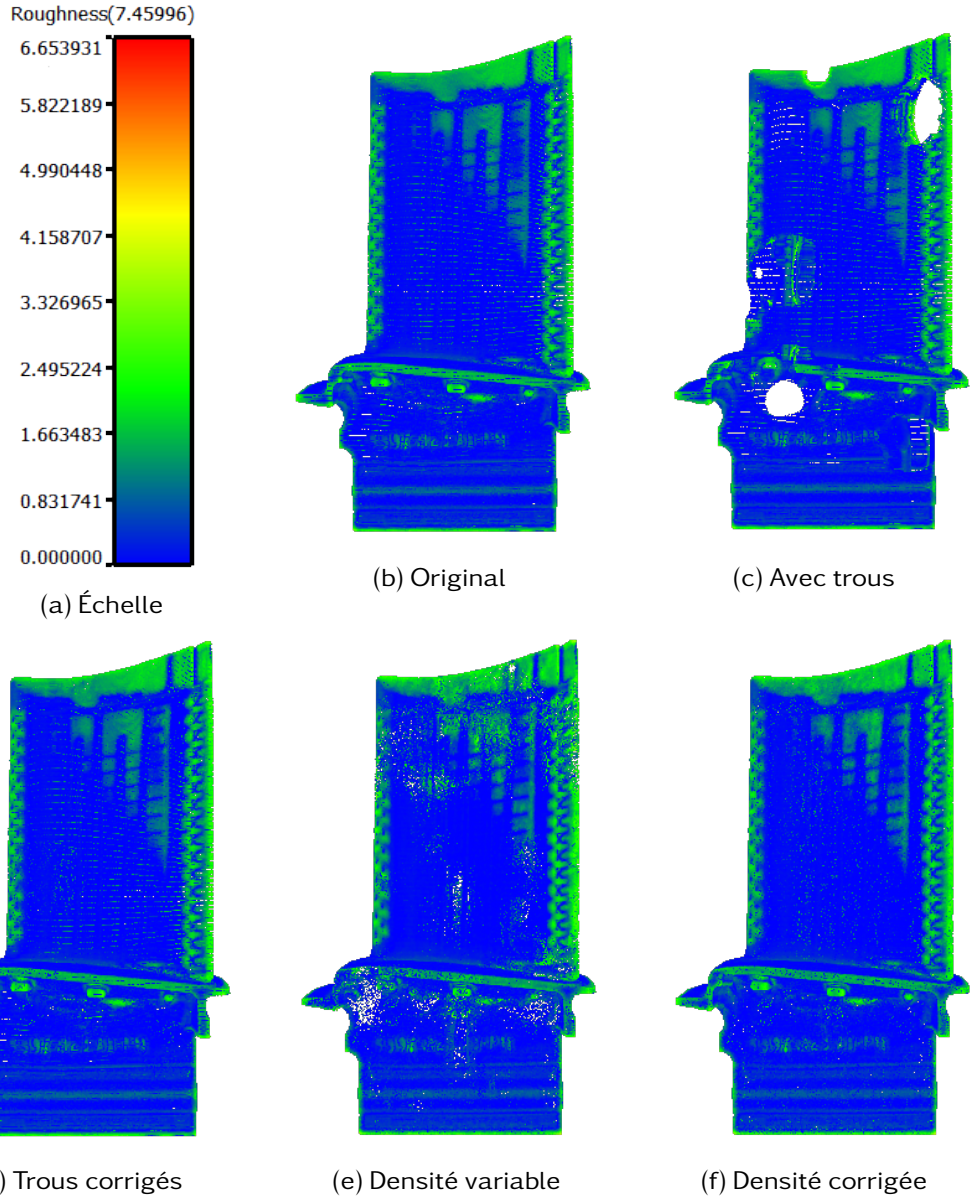


(e) Densité variable



(f) Densité corrigée

FIGURE 5.14 – Densité des modèles de pale (calculée par CloudCompare [clo])



(a) Échelle (b) Original (c) Avec trous

(d) Trous corrigés (e) Densité variable (f) Densité corrigée

FIGURE 5.15 – Rugosité des modèles de pale (calculée par CloudCompare [clo])

5.6 Influence de la taille des cellules et des répétitions

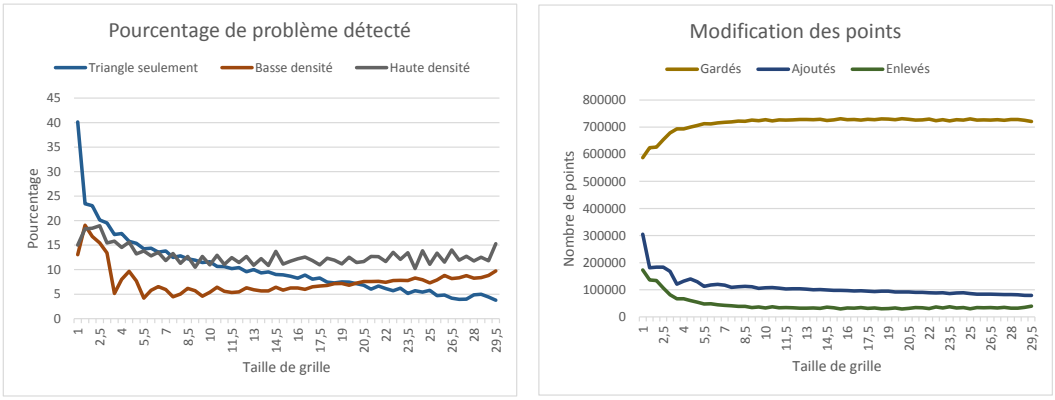


FIGURE 5.16 – Influence de la taille de grille

Nous l'avons dit, la taille de la grille est un paramètre important. Pour le vérifier, nous avons testé les corrections faîtes sur le modèle de turbine avec trous utilisé dans la section précédente dans un vaste intervalle de taille de cellule possible. Comme la figure 5.16 le montre, il en ressort que la taille choisie influe en effet beaucoup sur les résultats. Une taille trop petite va entraîner une trop grande détection de problème, résultant dans une suppression et un ajout de points trop importants. À l'inverse, une taille trop grande ne permet pas de détecter les trous et résulte en de mauvaises corrections, avec des artefacts fortement visibles. Nous pouvons remarquer que plus la taille de cellule augmente, moins il y a de trous détectés et plus de zones à faible densité, ce qui est logique. Cependant, cela veut aussi dire que les corrections seront moins ciblées dans les trous, résultant en une correction moins réussie.

Ce qui est rassurant en revanche, c'est qu'il existe un vaste plateau de valeurs entre les deux extrêmes où les résultats sont équivalents. Il est donc important de choisir une bonne taille de cellule, mais elle n'est pas trop difficile à trouver.

Nous avons également testé ce qui se passe en cas de répétition de l'application du processus avec les mêmes paramètres. Cela permet d'une part de vérifier que les corrections faites sont appropriées, mais également de savoir si les répétitions peuvent aider à obtenir un modèle plus proche de la qualité optimale qu'il est possible d'obtenir avec les modèles fournis. Pour le test, nous avons choisi de ne garder que la taille de cellule 5, qui donne de bons résultats dans le test réalisé ci-avant.

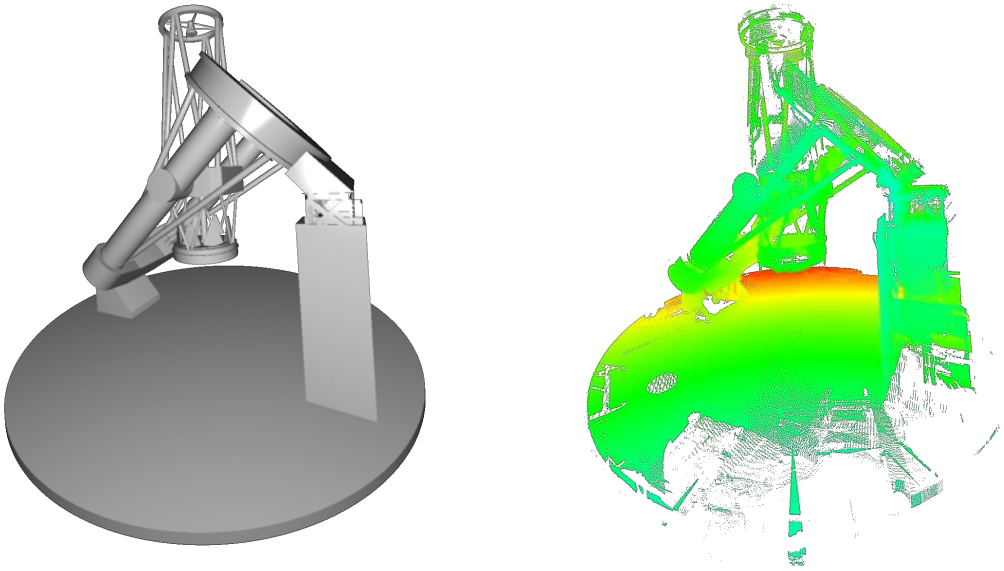
FIGURE 5.17 – Influence de l'itération

Le résultat est que pour ces deux aspects, la réponse est positive, comme le montre la figure 5.17. Attention, sur le graphique de droite l'échelle est logarithmique pour mieux voir les variations.

Nous pouvons en effet remarquer une forte chute des zones à problèmes dès la deuxième itération. Cependant une petite partie reste problématique, probablement due à la nature aléatoire de l'échantillonnage des triangles. Les problèmes détectés deviennent rapidement anecdotiques cependant, car moins de 10 000 points sont modifiés dès la deuxième itération, une valeur négligeable comparée aux plus de 800 000 points du modèle. Répéter le processus plusieurs fois sur le même modèle n'est donc pas nécessaire, mais peut permettre de forcer à respecter les contraintes un peu plus en cas de nécessité.

5.7 Résultats sur notre cas d'étude : le TBL

Après avoir vérifié que le processus et son implémentation fonctionnent en utilisant un modèle de test, cette section montre le traitement effectué sur un nuage de points réel provenant d'un scanner. L'objet représenté par ce modèle est le TBL [tbl]. Il se trouve au sommet du pic du Midi de Bigorre, dans les Hautes-Pyrénées. Le nuage de points provenant du scanner (figure 5.18b), possédant des défauts, ainsi qu'un maillage simple (figure 5.18a) sont utilisés en tant que données d'entrée pour l'application. Le but est donc de corriger le nuage de points à l'aide du maillage. Pour ce test, le maillage est supposé complet mais peu détaillé, alors que le nuage de points est incomplet mais bruité.



(a) Maillage (b) Nuage de points

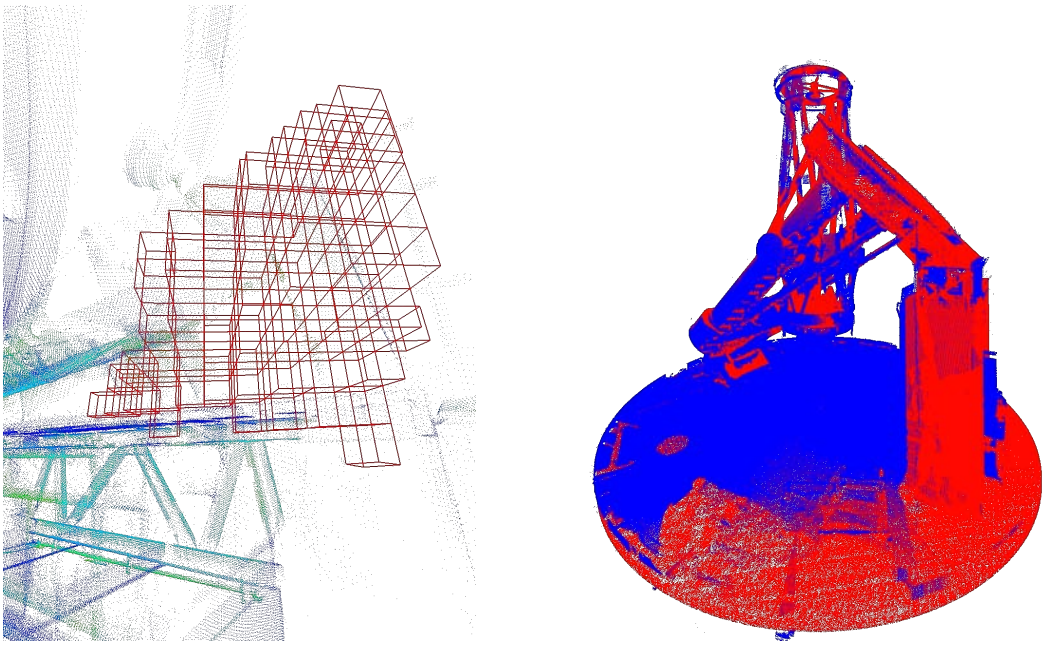
FIGURE 5.18 – Les modèles du télescope Bernard Lyot [tbl]

Le résultat est visible sur la figure 5.19b, avec en rouge les points ajoutés et en bleu les points originaux gardés. Un exemple de zone problématique tel que représenté dans l'application de test est montré figure 5.19a. Les couleurs de cette figure représentent la distance des points par rapport au maillage ; la zone problématique est représentée par des cubes rouges qui correspondent aux cellules de la grille régulière. Le problème de cette zone est qu'elle ne contient que des triangles (non représentés sur la figure).

	Le modèle du TBL
Nombre de points (millions)	11
Faible densité (%)	20
Haute densité (%)	8
Trous (%)	50
Pas de triangle (%)	20

TABLE 5.3 – Analyse du TBL

Au total, l'application a détecté 47 zones problématiques différentes, laissant environ 5% du modèle multi-représentations classifié comme sans problème. 8 de ces zones ne contiennent pas de points (comme sur la figure 5.19a), représen-



(a) Zoom sur une zone problématique (b) Résultat

FIGURE 5.19 – Le modèle traité

tant environ 50% du volume total de la grille. 13 zones ne contiennent pas de triangles, comptant pour 20%. Au niveau de la densité, environ 20% a été classé en zone de faible densité (20 zones), et 8% avec une densité trop élevée (6 zones). Un résumé est visible table 5.3.

Le modèle original contenait environ 11 millions de points, et environ 17 millions ont été ajoutés (en correspondance avec la densité calculée). Le nuage de points résultant du processus est clairement visuellement plus attirant. La partie manquante de la base visible en bas de la figure 5.18b a été complétée, ainsi que les piliers. Contrairement au modèle de test de la section 5.5.1, il n'est pas possible de distinguer la différence entre les points originaux et les points rajoutés sur ce modèle, car il n'y a pas de motif particulier. Le résultat est un nuage de points beaucoup plus adapté à la visualisation mais conservant les détails originaux que le modèle d'entrée possédait.

5.8 Synthèse et perspectives

Dans ce chapitre nous nous sommes intéressés à améliorer un nuage de points obtenu par un scanner, à l'aide d'un maillage. Nous avons commencé par détailler les problèmes qu'il est possible de corriger à l'aide des deux représentations : supprimer le bruit, compléter les trous et uniformiser la densité. Après avoir montré les capacités de l'application de test sur des cas triviaux et sur un cas réaliste contrôlé, nous avons montré les bons résultats obtenus sur un modèle issu d'un scanner laser.

De nombreuses perspectives sont ouvertes pour cette approche. La première, évidente, est d'être capable de traiter plus de représentations dans notre processus, en entrée et en sortie (voxels, maillage, etc.). Des possibilités d'améliorations sont également disponibles dans le cadre choisi dans ce chapitre. Le choix d'une grille régulière par exemple pour détecter les zones communes a été fait pour sa simplicité mais limite également la précision. Il serait donc intéressant de pouvoir utiliser des zones de forme libre, directement correspondantes aux modèles d'entrée au lieu de cellules. Détecter un décalage entre le nuage de points et le maillage serait aussi utile pour appliquer ce même décalage lors de l'ajout de points. La même idée pourrait être appliquée aux motifs, les détecter permettrait d'ajouter les points de la même manière et donc d'éviter des artefacts visuels. Enfin, au lieu d'ajouter des points globalement et de jouer avec les probabilités pour les placer là où c'est nécessaire, il pourrait être préférable d'agir localement pour être sûr que les problèmes soient bien réglés, même si nous avons vu que l'approche globale utilisée dans ce chapitre fonctionne bien.

5.9 Pour aller plus loin

Les détails ici ont été donnés pour corriger les nuages de points à l'aide de maillages. Mais comme dit dans la section précédente notre approche peut être utilisée avec de nombreuses sources et représentations de modèles 3D. Ce chapitre a présenté le concept et un cas d'application, le but n'étant pas de fournir des exemples sur toutes les combinaisons possibles. Cette section donne un exemple en plus, en réfléchissant à l'implémentation du processus dans le cas de l'amélioration du maillage à partir de nuages de points.

5.9.1 Déterminer les problèmes possibles

La première étape pour implémenter le processus est de déterminer les problèmes qu'il est possible de rencontrer. Des problèmes similaires à l'amélioration de nuages de points se rencontrent alors :

— les zones avec uniquement des triangles, qu'il faudra choisir de conserver ou supprimer,

— et les zones avec uniquement des points, qu'il faudra choisir de supprimer ou de trianguler sans indice sur la surface représentée.

Un troisième problème existe, légèrement différent du problème de densité : le niveau de détails. Il est en effet possible de supposer que le nuage de points contienne des détails que le maillage ne possède pas. Il faudra alors les transposer au maillage.

5.9.2 Détecter les problèmes possibles

La détection de zones communes est identique, aucune nouvelle difficulté apparaît ici. Il est donc aussi simple de détecter les zones contenant les deux premiers problèmes possibles cités ci-avant. La détection de zones où des détails manquent est en revanche plus difficile. Une première condition simple pour détecter de telles zones est de calculer la densité de points. Si elle est élevée, il est fortement possible que ce soit une zone de détails. En revanche elle n'est pas suffisante, car il est tout à fait possible qu'une zone complètement plate possède de nombreux points.

Deux voies s'ouvrent alors, si le nombre de triangles est un problème (s'il ne l'est pas, il est possible de vouloir trianguler tout le nuage de points avec le maillage en tant qu'indice pour trouver la surface). La première est peut être la plus simple, c'est de juste transposer toutes les zones à forte densité de points pour ensuite appliquer un algorithme de simplification au maillage, supprimant tous les triangles inutiles. La deuxième est de pousser la détection plus loin pour être sûr de ne transposer que les zones contenant des détails. Une manière de faire cela pourrait être de détecter si tous les points sont sur un même plan, les plans utilisés pouvant être fournis par le maillage. Si c'est le cas, nous pouvons alors supposer que c'est seulement une zone contenant beaucoup de point, autrement c'est une zone de détails qui manquent au maillage.

5.9.3 Corriger les problèmes possibles

Le problème principal est l'ajout de détails. Une approche possible serait de trianguler la partie qui nous intéresse, en utilisant comme indice les polygones proches du maillage pour retrouver la surface. Une fois ce nouveau maillage obtenu, il faut le fusionner avec le maillage, avec par exemple les algorithmes de [Agugiaro & Kolbe 2012] ou de [Lou *et al.* 2010]. À la fin de ces corrections, le modèle maillé obtenu comportera les détails qui manquaient.

Chapitre 6

Conclusion

Avec la démocratisation des scanners lasers et des modèles en nuage de points 3D qu'ils génèrent, cette thèse a eu pour but de corriger les défauts obtenus pendant l'acquisition de ces modèles. Elle a pour cela présenté un processus en quatre étapes.

La première étape est une étape de sélection, ayant pour but de ne garder que les parties possédant un intérêt particulier pour l'application courante. Bien que des techniques pour cela existaient déjà, elles comportaient des lacunes, ce qui nous a amené à proposer une nouvelle technique. En effet, les techniques de l'état de l'art, bien que remplissant leur fonction, posent problème pour sélectionner des formes complexes et ne rendent pas la chose aisée pour un utilisateur. Notre technique a donc été créée afin de rendre ces sélections complexes plus simples, tout en gardant les sélections simples de même difficulté. Les résultats de nos expériences utilisateurs ont montré que cette technique pouvait être la plus performante parmi celles testées, tirées de l'état de l'art, mais surtout était préférée par nos utilisateurs. Des améliorations peuvent être effectuées afin d'améliorer les performances tout comme la prise en main par un utilisateur. Cette technique représente néanmoins un bon marchepied vers d'autres techniques de sélection innovantes tirant parti des améliorations présentées ici.

Les étapes suivantes détaillent ensuite l'analyse et la correction des problèmes possibles d'un modèle. Là encore des améliorations peuvent améliorer la technique. Tout d'abord une meilleure détection des zones communes peut être réalisée, en utilisant un découpage dynamique comme un *octree* d'une part, qui va s'adapter localement aux modèles 3D, ou par la définition plus précise d'une zone d'autre part, sans s'appuyer uniquement sur des délimitations cubiques qui limitent les formes détectées. Ensuite, une amélioration peut éventuellement être faite sur la détection de zones à problème, en définissant et détectant des cas non

prévus actuellement, s'ils existent. Enfin, il est possible d'utiliser une approche différente pour la correction des problèmes, en s'appuyant sur une approche un peu moins aléatoire de l'échantillonnage utilisé actuellement par exemple.

De plus, le processus dans son ensemble n'a été testé que dans le cas de l'amélioration de nuages de points et sur peu de modèles. Des travaux complémentaires peuvent être menés dans cette direction également. Le tester sur plus de modèles permettrait de renforcer la preuve de son utilité, mais pourrait aussi dévoiler des faiblesses à corriger. Des travaux futurs peuvent également se diriger vers la prise en charge de plus de représentations, en commençant en premier lieu la démarche de déterminer les problèmes possibles et la façon de les corriger, pour ensuite intégrer la détection et la résolution de ces problèmes dans le processus.

Cependant, cette première version de l'implémentation du processus montre déjà que l'idée de base fonctionne très bien et donne des résultats plus que satisfaisants visuellement. Le potentiel est ainsi démontré, ce qui était le but de ces travaux. Utiliser des données complémentaires contenant des informations sur un objet permet de réparer la représentation d'un modèle 3D et d'obtenir en résultat un modèle plus proche de la perfection.

Bibliographie

[Agugiaro & Kolbe 2012] Giorgio Agugiaro et Thomas H. Kolbe. *A deterministic method to integrate triangular meshes of different resolution*. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 71, pages 96–109, 2012. (Cité en page 112.)

[Alexa *et al.* 2001] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin et Claudio T. Silva. *Point Set Surfaces*. In Proceedings of the Conference on Visualization '01, VIS '01, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society. (Cité en page 37.)

[Bangor *et al.* 2008] Aaron Bangor, Philip T. Kortum et James T. Miller. *An Empirical Evaluation of the System Usability Scale*, 2008. (Cité en pages 78 et 79.)

[Bendels *et al.* 2006] G.H. Bendels, R. Schnabel et Reinhard Klein. *Detecting Holes in Point Set Surfaces*. The Journal of WSCG, vol. 14, 2006. (Cité en pages xiv, 42 et 43.)

[Benko & Feiner 2007] Hrvoje Benko et Steven Feiner. *Balloon selection : A multi-finger technique for accurate low-fatigue 3D selection*. In IEEE Symposium on 3D User Interfaces 2007 - Proceedings, 3DUI 2007, pages 79–86. IEEE, 2007. (Cité en pages xiii, 27, 28 et 73.)

[Bergé *et al.* 2014] Louis-Pierre Bergé, Gary Perelman, Adrien Hamelin, Mathieu Raynal, Cédric Sanza, Minica Houry-Panchetti, Rémi Cabanac et Emmanuel Dubois. *Smartphone Based 3D Navigation Techniques in an Astronomical Observatory Context : Implementation and Evaluation in a Software Platform*. International Journal on Advances in Software, Extended version of a best paper award contribution published at the Seventh International Conference on Advances in Computer-Human Interactions (ACHI 2014), vol. 7, no. 3 & 4, pages 551–566, décembre 2014. Extended version of a best paper award contribution published at the Seventh International Conference on Advances in Computer-Human Interactions (ACHI 2014). (Cité en pages 7 et 18.)

[Blender Foundation 1995] Blender Foundation. *Blender*. http://www.blender.org/, 1995. (Cité en pages 11 et 30.)

[Bowman *et al.* 2001] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola et Ivan Poupyrev. *An Introduction to 3-D User Interface Design*. Presence : Teleoper. Virtual Environ., vol. 10, no. 1, pages 96–108, Février 2001. (Cité en pages 18 et 20.)

[Brodsky & Watson 2000] Dmitry Brodsky et Benjamin Watson. *Model simplification through refinement*. In Graphics Interface, volume 2000, pages 221–228, 2000. (Cité en page 37.)

[Brooke 1996] John Brooke. *SUS-A quick and dirty usability scale*. Usability evaluation in industry, vol. 189, page 194, 1996. (Cité en pages 72 et 78.)

[Cabral *et al.* 2014] Marcio Cabral, Andre Montes, Olavo Belloc, Rodrigo Ferraz, Fernando Teubl, Fabio Doreto, Roseli Lopes et Marcelo Zuffo. *Bi-manual gesture interaction for 3D cloud point selection and annotation using COTS*. In 2014 IEEE Symposium on 3D User Interfaces (3DUI), pages 187–188. IEEE, Mars 2014. (Cité en pages 27 et 73.)

[Cashion & LaViola 2014] Jeffrey Cashion et Joseph J LaViola. *Poster : Dynamic adaptation of 3D selection techniques for suitability across diverse scenarios*. In 2014 IEEE Symposium on 3D User Interfaces (3DUI), pages 165–166. IEEE, Mars 2014. (Cité en page 26.)

[clo] *CloudCompare*. http://www.danielgm.net/cc/. (Cité en pages 74, 104 et 105.)

[Company 2013] The Fullbright Company. *Gone home*. http://www.gonehomegame.com/, 2013. (Cité en page 18.)

[Digne 2012] Julie Digne. *Similarity based filtering of point clouds*. In 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 73–79. IEEE, Juin 2012. (Cité en pages xiv et 36.)

[Doria & Radke 2012] David Doria et Richard J. Radke. *Filling large holes in LiDAR data by inpainting depth gradients*. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 65–72, 2012. (Cité en pages xiv et 44.)

[Elmqvist & Fekete 2008] Niklas Elmqvist et Jean-Daniel Fekete. *Semantic Pointing for Object Picking in Complex 3D Environments*. In Proceedings of Graphics Interface 2008, GI '08, pages 243–250, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society. (Cité en page 20.)

[Gois *et al.* 2006] J.P. Gois, Eduardo Tejada, Tiago Etiene, L.G. Nonato, Antonio Castelo et Thomas Ertl. *Curvature-driven modeling and rendering of point-based surfaces*. In 2006 19th Brazilian Symposium on Computer Graphics and Image Processing, pages 27–36. IEEE, Octobre 2006. (Cité en page 13.)

[Hamelin & Dubois 2015] Adrien Hamelin et Emmanuel Dubois. *Design and evaluation of an interaction technique for volume selection in a 3D point cloud*. In 27ème conférence francophone sur l'Interaction Homme-Machine., page a3, Toulouse, France, Octobre 2015. ACM. (Cité en page 55.)

[He & Cheng 2011] Guizhen He et Xiaojun Cheng. *A Virtual Restoration Strategy of 3D Scanned Objects*. In Sally Jin, David and Lin, editeur, Advances in Computer Science, Intelligent System and Environment, pages 621–627. Springer Berlin Heidelberg, 2011. (Cité en pages xiv, 43 et 44.)

[Hou *et al.* 2012] Wenguang Hou, Taiwai Chan et Mingyue Ding. *Denoising point cloud*. Inverse Problems in Science and Engineering, vol. 20, no. 3, pages 287–298, Avril 2012. (Cité en pages xiv et 35.)

[Huang *et al.* 2013] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher et Hao (Richard) Zhang. *Edge-aware point set resampling*. ACM Transactions on Graphics, vol. 32, no. 1, pages 1–12, Janvier 2013. (Cité en pages xiv, 38 et 39.)

[Jankowski & Hachet 2013] Jacek Jankowski et Martin Hachet. *A Survey of Interaction Techniques for Interactive 3D Environments*. In Eurographics 2013 - STAR, Girona, Espagne, 2013. (Cité en pages xiii, 18 et 19.)

[Jin *et al.* 2006] Xiaogang Jin, Juncong Lin, Charlie C.L. Wang, Jieqing Feng et Hanqiu Sun. *Mesh fusion using functional blending on topologically incompatible sections*. The Visual Computer, vol. 22, no. 4, pages 266–275, 2006. (Cité en page 42.)

[Kanai *et al.* 1999] Takashi Kanai, Hiromasa Suzuki, Jun Mitani et Fumihiko Kimura. *Interactive Mesh Fusion Based on Local 3D Metamorphosis*. In Proceedings of the 1999 Conference on Graphics Interface '99, pages 148–156, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. (Cité en page 42.)

[Kang *et al.* 2013] Jinsheng Kang, Kang Zhong, Shengfeng Qin, Hongan Wang et David Wright. *Instant 3D design concept generation and visualization by*

real-time hand gesture recognition. Computers in Industry, vol. 64, no. 7, pages 785–797, Septembre 2013. (Cité en pages 31 et 32.)

[Katz *et al.* 2007] Sagi Katz, Ayellet Tal et Ronen Basri. *Direct visibility of point sets*. ACM Transactions on Graphics, vol. 26, no. 3, page 24, Juillet 2007. (Cité en pages 14 et 21.)

[Kirkvik 2008] Ann-Silje Kirkvik. Completing a model based on laser scan generated point cloud data. Master's thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2008. (Cité en page 42.)

[Krammes *et al.* 2014] Hernandi Krammes, Marcio M. Silva, Theodoro Mota, Matheus T. Tura, Anderson Maciel et Luciana Nedel. *The point walker multi-label approach*. In IEEE Symposium on 3D User Interfaces 2014, 3DUI 2014 - Proceedings, pages 189–190. IEEE, Mars 2014. (Cité en pages xiv, 29, 31 et 73.)

[Labatut 2009] Patrick Labatut. *Labeling of data-driven complexes for surface reconstruction*. Theses, Université Paris-Diderot - Paris VII, Septembre 2009. (Cité en page 41.)

[Lee *et al.* 2003] Sangyoon Lee, Jinseok Seo, Gerard Jounghyun Kim et Chan-mo Park. *Evaluation of pointing techniques for ray casting selection in virtual environments*. In In Third International Conference on Virtual Reality and Its Application in Industry, pages 38–44, 2003. (Cité en page 25.)

[Liu *et al.* 2013] Zhen-Bao Liu, Shu-Hui Bu, Kun Zhou, Shu-Ming Gao, Jun-Wei Han et Jun Wu. *A Survey on Partial Retrieval of 3D Shapes*. Journal of Computer Science and Technology, vol. 28, no. 5, pages 836–851, Septembre 2013. (Cité en pages xiv et 40.)

[Lou *et al.* 2010] Ruding Lou, Jean-Philippe Pernot, Alexei Mikchevitch et Philippe Véron. *Merging enriched Finite Element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance*. Computer-Aided Design, vol. 42, no. 8, pages 670 – 681, 2010. Application-driven Shape Development. (Cité en page 112.)

[Lucas *et al.* 2005] John F Lucas, Doug Bowman, Jian Chen et Chadwick A Wingrave. *Design and Evaluation of 3D Multiple Object Selection Techniques*. In Proc. of the ACM I3D, 2005. (Cité en pages xiii, 26, 28 et 73.)

[Ma *et al.* 2010] Teng Ma, Zhuangzhi Wu, Lu Feng, Pei Luo et Xiang Long. *Point cloud segmentation through spectral clustering*. In The 2nd International Conference on Information Science and Engineering, pages 1–4. IEEE, Décembre 2010. (Cité en pages xiii, 22 et 23.)

[Mackinlay *et al.* 1990] Jock D. Mackinlay, Stuart K. Card et George G. Robertson. *Rapid Controlled Movement Through a Virtual 3D Workspace*. SIGGRAPH Comput. Graph., vol. 24, no. 4, pages 171–176, Septembre 1990. (Cité en page 18.)

[Naito *et al.* 2009] Masaki Naito, Buntarou Shizuki, Jiro Tanaka et Hiroshi Hosobe. *Interaction techniques using a spherical cursor for 3d targets acquisition and indicating in volumetric displays*. In Proceedings of the International Conference on Information Visualisation, pages 607–612. IEEE, Juillet 2009. (Cité en pages xiii, 28 et 29.)

[Nan *et al.* 2012] Liangliang Nan, Ke Xie et Andrei Sharf. *A search-classify approach for cluttered indoor scene understanding*. ACM Transactions on Graphics, vol. 31, no. 6, page 1, Novembre 2012. (Cité en pages xiii, 23 et 24.)

[Nielson & Olsen 1987] Gregory M. Nielson et Dan R. Olsen Jr. *Direct Manipulation Techniques for 3D Objects Using 2D Locator Devices*. In Proceedings of the 1986 Workshop on Interactive 3D Graphics, I3D '86, pages 175–182, New York, NY, USA, 1987. ACM. (Cité en page 20.)

[of Technology] Georgia Institute of Technology. *Turbine blade model*. `http://www.cc.gatech.edu/projects/large_models/blade.html`. (Cité en page 99.)

[Panchetti *et al.* 2010] Minica Panchetti, Jean-Philippe Pernot et Philippe Véron. *Towards recovery of complex shapes in meshes using digital images for reverse engineering applications*. Computer-Aided Design, vol. 42, no. 8, pages 693 – 707, 2010. Application-driven Shape Development. (Cité en page 47.)

[Park *et al.* 2013] Min Ki Park, Seung Joo Lee, In Yeop Jang, Yong Yi Lee et Kwan H. Lee. *Feature-aware filtering for point-set surface denoising*. Computers & Graphics, vol. 37, no. 6, pages 589–595, Octobre 2013. (Cité en pages xiv, 36 et 37.)

[Pauly *et al.* 2002] M. Pauly, M. Gross et L.P. Kobbelt. *Efficient simplification of point-sampled surfaces*. In IEEE Visualization, 2002. VIS 2002., VIS '02, pages 163–170, Washington, DC, USA, 2002. IEEE. (Cité en pages xiv, 37 et 38.)

[Pernot *et al.* 2006] Jean-Philippe Pernot, George Moraru et Philippe Véron. *Filling holes in meshes using a mechanical model to simulate the curvature variation minimization*. Computers & Graphics, vol. 30, no. 6, pages 892 – 902, 2006. (Cité en page 43.)

[Pixar 1995] Pixar. *Toy Story*. http://www.pixar.com/features_films/TOY-STORY, 1995. (Cité en pages 3 et 19.)

[Pixologic] Pixologic. *ZBrush*. http://pixologic.com/zbrush/features/overview/. (Cité en page 16.)

[Projekt 2015] CD Projekt. *The Witcher 3*. http://thewitcher.com/witcher3/, 2015. (Cité en page 11.)

[Reuter *et al.* 2010] Patrick Reuter, Guillaume Riviere, Nadine Couture, Stephanie Mahut et Loic Espinasse. *ArcheoTUI&Mdash ;Driving Virtual Reassemblies with Tangible 3D Interaction*. J. Comput. Cult. Herit., vol. 3, no. 2, pages 4 :1–4 :13, Octobre 2010. (Cité en page 68.)

[Sithole & Vosselman 2003] G. Sithole et G. Vosselman. *Automatic structure detection in a point-cloud of an urban landscape*. In Remote Sensing and Data Fusion over Urban Areas, 2003. 2nd GRSS/ISPRS Joint Workshop on, pages 67–71, May 2003. (Cité en page 14.)

[Sowell *et al.* 2009] R Sowell, L Liu, T Ju, C Grimm, C Abraham, G Gokhroo et D Low. *Volume viewer : an interactive tool for fitting surfaces to volume data*. In SBIM '09 : Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling, pages 141–148, New York, NY, USA, 2009. ACM. (Cité en pages xiv, 32 et 33.)

[Stamati & Fudos 2007] Vasiliki Stamati et Ioannis Fudos. *A feature based approach to re-engineering objects of freeform design by exploiting point cloud morphology*. In Proceedings of the 2007 ACM symposium on Solid and physical modeling - SPM '07, SPM '07, page 347, New York, New York, USA, 2007. ACM Press. (Cité en pages xiii et 22.)

[Steinicke *et al.* 2004] Frank Steinicke, Timo Ropinski et Klaus Hinrichs. *Object selection in virtual environments with an improved virtual pointer metaphor*.

Symposium A Quarterly Journal In Modern Foreign Literatures, pages 320–326, 2004. (Cité en page 25.)

[Systèmes a] Dassault Systèmes. *Catia*. http://www.3ds.com/fr/produits-et-services/catia. (Cité en page 17.)

[Systèmes b] Dassault Systèmes. *SolidWorks*. http://www.solidworks.fr/. (Cité en page 17.)

[tbl] *Télescope Bernard Lyot*. http://www.tbl.omp.eu/. (Cité en pages xvi, 55, 107 et 108.)

[Turk 1992] Greg Turk. *Re-tiling Polygonal Surfaces*. SIGGRAPH Comput. Graph., vol. 26, no. 2, pages 55–64, Juillet 1992. (Cité en page 38.)

[Ulinski *et al.* 2007] Amy Ulinski, Catherine Zanbaka, Zachary Wartell, Paula Goolkasian et Larry F. Hodges. *Two handed selection techniques for volumetric data*. In IEEE Symposium on 3D User Interfaces 2007 - Proceedings, 3DUI 2007, pages 107–114, 2007. (Cité en pages xiv, 29 et 30.)

[Veit & Capobianco 2014] Manuel Veit et Antonio Capobianco. *Go'Then'Tag : A 3-D point cloud annotation technique*. In 2014 IEEE Symposium on 3D User Interfaces (3DUI), pages 193–194. IEEE, Mars 2014. (Cité en pages xiii et 25.)

[Vinayak *et al.* 2013] Vinayak, Sundar Murugappan, Hairong Liu et Karthik Ramani. *Shape-It-Up : Hand gesture based creative expression of 3D shapes using intelligent generalized cylinders*. In CAD Computer Aided Design, volume 45, pages 277–287, Février 2013. (Cité en pages xiv, 32 et 58.)

[Wang & Shan 2009] Jun Wang et Jie Shan. *Segmentation of LiDAR point clouds for building extraction*. In American Society for Photogramm. Remote Sens. Annual Conference, Baltimore, MD, pages 9–13, 2009. (Cité en page 21.)

[Wang *et al.* 2007] Renfang Wang, Sanyuan Zhang et Xiuzi Ye. *A novel simplification algorithm for point-sampled surfaces*. In 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), pages 573–578. IEEE, 2007. (Cité en pages xiv et 38.)

[Yu *et al.* 2011] Zhiding Yu, Chunjing Xu, Jianzhuang Liu, Oscar C Au et Xiaoou Tang. *Automatic object segmentation from large scale 3D urban point clouds through manifold embedded mode seeking*. In Proceedings of the 19th {ACM} international conference on Multimedia, MM '11, pages 1297–1300, New York, New York, USA, 2011. ACM Press. (Cité en pages xiii et 23.)

[Yu *et al.* 2012] Lingyun Yu, Konstantinos Efstathiou, Petra Isenberg et Tobias Isenberg. *Efficient structure-aware selection techniques for 3D point cloud visualizations with 2DOF input*. IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 12, pages 2245–2254, 2012. (Cité en page 26.)

Glossaire

CAO

conception assistée par ordinateur. *Glossaire :* conception assistée par ordinateur, 9, 14, 123

conception assistée par ordinateur

Terme généralement utilisé pour la création d'objets de forme libre. 9, 123

ECP

Enveloppe de Contours Parallèles. *Glossaire :* Enveloppe de Contours Parallèles, 72, 73, 123

Enveloppe de Contours Parallèles

Technique de sélection par volume de l'état de l'art, voir chapitre 4. 72, 123

Enveloppe de Sélection Adaptative

Nouvelle technique de sélection dans un nuage de points 3D proposée par ce manuscrit, décrite dans le chapitre 4. xiv, 55, 123

ESA

Enveloppe de Sélection Adaptative. *Glossaire :* Enveloppe de Sélection Adaptative, xiv, 55, 56, 57, 58, 59, 60, 61, 68, 72, 73, 74, 76, 77, 78, 79, 80, 81, 123

lancer de rayon

Technique qui consiste à simuler le lancer d'un rayon dans un environnement 3D. Cette technique peut être utilisée pour pointer vers un objet, mais également pour faire un rendu 3D de qualité. 24, 25, 33, 56

maillage

Un modèle 3D composé de polygones représentants sa surface, le plus souvent des triangles. 9, 12, 14, 15, 16, 17, 24, 25, 42, 48, 59, 60, 83, 84, 85, 86, 87, 90, 95, 98, 99, 101, 102, 107, 109, 110, 111

marching-cube

Algorithme permettant de convertir une fonction implicite en un maillage. Il a l'avantage d'être simple à utiliser et offre des résultats corrects. 41

mean-shift

Algorithme permettant de trouver le maximum d'une fonction. 23

min-cut

Algorithme dont le but est de découper un graphe en minimisant un critère, comme par exemple le nombre d'arêtes coupées. 22

normale

Un vecteur unitaire orthogonal à la surface d'un modèle 3D ayant pour origine une primitive comme un point ou un triangle. 12, 22, 36, 59, 63

nuage de points

Un modèle 3D composé de points représentants sa surface. i, 6, 7, 9, 10, 12, 14, 15, 17, 19, 21, 22, 23, 25, 26, 33, 35, 37, 41, 42, 44, 45, 48, 55, 56, 58, 60, 69, 79, 81, 83, 84, 85, 86, 87, 88, 89, 90, 91, 98, 99, 101, 103, 107, 109, 110, 111, 113, 114

objet de forme libre

Un modèle 3D composé de formes quelconques, non limitées (comme pour un maillage), généralement définies par un ensemble de formules mathématiques. 16, 17, 30, 31, 33, 57, 59

Observatoire Midi Pyrénées

Laboratoire scientifique dont les recherches sont dans le domaine de l'astronomie. 7, 124

octree

Une structure de donnée divisant chacune de ses cellules en 8 autres jusqu'à un critère d'arrêt, comme une profondeur maximum ou un nombre d'élément dans une cellule. 25, 44, 60, 113

OMP

Observatoire Midi Pyrénées. *Glossaire :* Observatoire Midi Pyrénées, 7, 9, 124

Sélection par Formes Prédéfinies

Technique de sélection par volume de l'état de l'art, voir chapitre 4. 73, 124

SFP

Sélection par Formes Prédéfinies. *Glossaire :* Sélection par Formes Prédéfinies, 73, 74, 76, 77, 78, 79, 80, 81, 124

TBL

Télescope Bernard Lyot. *Glossaire :* Télescope Bernard Lyot, 6, 7, 9, 55, 101, 107, 124

Télescope Bernard Lyot

Télescope se trouvant au sommet du Pic du Midi. 6, 55, 124