Construction automatique de solides à partir de vues orthogonales de type dessin industriel

Rémi Lequette

THESE

Présentée à

L'UNIVERSITE SCIENTIFIQUE, TECHNOLOGIQUE ET MEDICALE DE GRENOBLE

pour obtenir le grade de
DOCTEUR de l'UNIVERSITE SCIENTIFIQUE,
TECHNOLOGIQUE ET MEDICALE DE GRENOBLE
spécialité : Mathématiques Appliquées
option : Recherche Opérationnelle

par

Rémi LEQUETTE

ooooo

CONSTRUCTION AUTOMATIQUE DE SOLIDES A PARTIR DE VUES ORTHOGONALES DE TYPE DESSIN INDUSTRIEL

ooooo

Thèse soutenue le 26 Octobre 1987 devant la commission d'examen.

J. MERMET	**Président**
J.M. BRUN	
J.P. CRESTIN	**Examinateurs**
B. LACOLLE	
J.P. UHRY	

UNIVERSITE SCIENTIFIQUE TECHNOLOGIQUE ET MEDICALE DE GRENOBLE

Président de l'Université :
M. TANCHE

Année Universitaire 1986 - 1987

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ère Classe

ARNAUD Paul	Chimie Organique
ARVIEU ROBERT	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARBIER Jean-Claude	Physique Expérimentale CNRS
BARJON Robert	Physique Nucléaire ISN
BARNOUD Fernand	Boichimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BELORISKY Elie	Physique C.E.N.G- D.R.F.
BENZAKEN Claude	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire ISN
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique`
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DELOBEL Claude	Mathématiques Appliquées
DEPORTES Charles	Chimie Minérale
DESRE Pierre	Electrochimie
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Rolland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean-Pierre	Mécanique,
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées
KUPKA Yvon	Mathématiques Pures
LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
DE LEIRIS Joel	Biologie

LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
MACHE Régis	Physiologie Végétale
MAYNARD Roger	Physique du Solide
MICHEL Robert	Minéralogie et Pétrographie (Géologie)
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIAUX Jacques	Géologie
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
PIERRE Jean-Louis	Chimie Organique
RASSAT André	Chimie Systématique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAKAROVITCH Michel	Mathématiques Appliquées
SAXOD Raimard	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BLANCHI J.Pierre	STAPS
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET J.François	Physique
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHARDON Michel	Géographie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPASSEL Roger	Mécanique des Fluides
DERRIEN Jacques	Physique
DUFREYNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
GUITTON Jacques	Chimie
HACQUES Gérard	Mathématiques Appliquées

HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKHOVE Claude	Géologie
LEBRETON Alain	Mathématiques Appliquées
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N.
LUCAS Robert	Physique
LUNA Domingo	Mathématiques Pures
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
MASCLE Georges	Géologie
NEMOZ Alain	Thermodynamique CNRS - CRTBT
OUDET Bruno	Mathématiques Appliquées
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean-Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RIEDIMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SOUTIF Jeanne	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L' IUT 1

PROFESSEURS de 1ere Classe

BUISSON Roger	Physique IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1

PROFESSEURS de 2ème classe

BOUTHINON Michel	EEA. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHEHIKIAN Alain	EEA. IUT 1
CHENAVAS Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
GOSSE Jean-Pierre	EEA.IUT 1
GROS Yves	Physique IUT 1
KUHN Gérard, (Détaché)	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	EEA.IUT 1
NOUGARET Marcel	Automatique IUT 1
PEFFEN René	Métallurgie IUT 1
PERARD Jacques	EEA. IUT 1
PERRAUD Robert	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
VINCENDON Marc	Chimie IUT 1

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXEPTIONNELLE ET 1ère CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puericulture	C.H.R.G.
BEZEZ Henri	Orthopédie-Traumatologie	Hopital SUD
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMPETIER Jean	Anatomie-Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COUDERC Pierre	Anatomie-Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologique	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Qbstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VERAIN André	Biophysique	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie-Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	ABIDJAN
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital Sud
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et Informatique Médicale	Faculté La Merci

DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie-Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie	C.H.R.G.
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

MEMBRES DU CORPS ENSEIGNANT PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
BOUCHERLE André	Chimie et Toxicologie	Faculté Meylan
CUSSAC Max	Chimie Therapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiéne et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

Remerciements

Je tiens tout d'abord à remercier Monsieur Jean Mermet, Directeur du laboratoire ARTEMIS, qui a eu la première initiative de ce travail, qui m'a accueilli dans son laboratoire et m'a fait l'honneur de présider le jury de cette thèse.

Je dois ensuite exprimer ma gratitude aux deux personnes qui ont suivi personellement ce travail : Messieurs Jean-Marc Brun, Directeur scientifique de Matra-Datavision et Jean-Pierre Uhry du laboratoire ARTEMIS, directeur de cette thèse.

Il me faut aussi remercier tout particulièrement Messieurs Jean-Pierre Crestin, Chef du groupe " Informatique et Automatique " à la DRET, et Bernard Lacolle de l'Université de Grenoble et du laboratoire TIM3, qui ont accepté d'être rapporteurs.

Je n'oublierai pas non plus les deux équipes de recherche qui m'ont accueillies : l'équipe " Recherche Opérationnelle " du laboratoire ARTEMIS-IMAG et l'équipe " Recherche Développement " de Matra-Datavision. J'ai trouvé dans ces deux équipes une ambiance de travail sympathique et chaleureuse.

Plusieurs organismes ont assuré le soutien de mon travail : le laboratoire ARTEMIS de l'IMAG (CNRS - Université de Grenoble I - Institut Polyechnique de Grenoble), Matra-Datavision et l'ANRT (Association Nationale pour la Recherche Technique).

Je tiens aussi à remercier Josiane CARRY pour son aide dans la production du rapport et les services de reproduction de l'IMAG pour le tirage.

Table

Introduction

La première étape de l'introduction des ordinateurs dans les bureaux d'études industriels n'a pas été une grande révolution. C'est celle des systèmes de D.A.O (Dessin Assisté par Ordinateur) qui ont permis d'automatiser complètement le travail du dessinateur industriel, mais sans changer fondamentalement celui des ingénieurs. Puis sont venus des systèmes tridimensionnels qui permettent de produire automatiquement plusieurs vues de la pièce étudiée. Ces systèmes ont pratiquement envahi toute l'industrie puisqu'on les trouve même sur les micro-ordinateurs le plus modestes.

La seconde étape du processus est l'arrivée de la véritable C.A.O (Conception Assisté par Ordinateur, ou même C.F.A.O pour Fabrication). Les systèmes de C.A.O encore réservés aux grands utilisateurs permettent de travailler sur un véritable modèle de la pièce à réaliser. A partir de ce modèle peuvent être développés tous les calculs et simulations qui interviennent dans le cycle de conception :

- Calculs de masse, volume, encombrement, interférences.

- Calculs de cinématique et de dynamique.

- Maillage et calculs d'éléments finis (résistance, contraintes, aérodynamique...)

- Conception des gammes de fabrication.

- Génération de programmes pour machines à commande numérique.

Avec ces systèmes on retrouve le véritable but du bureau d'étude qui est de concevoir des produits et non de faire des plans. Les représentations graphiques si elles sont indispensables à la communication sont en fait un bien pauvre modèle pour concevoir une pièce.

Le sujet de recherche présenté dans cette thèse est parti d'une motivation industrielle proposée par Monsieur Jean-Marc Brun de Matra-Datavision (fournisseur du système de C.F.A.O solide EUCLID). Ils s'agissait d'étudier des techniques pour interpréter automatiquement des plans de dessin industriel. Comme nous l'avons remarqué plus haut de nombreux industriels utilisent encore le plan comme seul support d'information. Il pourrait donc être très intéressant de disposer d'outils pour extraire le maximum d'information de ces plans et la mettre sous la forme requise par les systèmes de C.A.O solides. Une demande potentielle immense existe qui n'est pas toujours exprimée car aucune solution n'est encore proposée.

La première partie de ce rapport est une brève **Introduction au sujet** proprement dit. On présente plus en détail ce qu'est un système de C.A.O et les motivations qui ont conduit à ce travail. On propose une première analyse du problème pour le délimiter.

Le cadre de cette thèse est la **modélisation des solides** c'est à dire l'ensemble des techniques mises en oeuvre pour modéliser dans une structure de donnée informatique des solides tridimensionnels. La deuxième partie est donc entièrement consacrée à cette modélisation des solides. On y trouvera les définitions et les principes théoriques fondamentaux ainsi que les différentes techniques utilisées aujourd'hui. Nous insisterons particulièrement sur la représentation surfacique d'un solide qui le décompose en **faces**, **arêtes** et **sommets**.

Dans la troisième partie nous nous pencherons sur une **approche purement topologique** du problème. C'est à dire que nous ne ferons pas intervenir de calculs géométriques, mais uniquement les propriétés topologiques ou combinatoires des graphes sous-jacents aux solides. On verra que nous n'avons pas retenu cette approche pour la solution que nous proposons, cependant elle apporte des informations intéressantes et indispensables pour une bonne compréhension des structures

mathématiques du problème. L'idée de cette approche est de considérer le graphe formé par les arêtes et les sommets d'un solide et d'appliquer des résultats connus sur les différentes manières de dessiner un graphe sur une surface.

Dans la quatrième partie, nous proposons notre **solution au problème de reconstruction de solides à partir de plans**. Cette solution se décompose en deux étapes qui sont :

- Construire à partir des plans un modèle en trois dimensions formé de sommets et d'arêtes appelé modèle **fil-de-fer**.

- Construire un ou plusieurs solides à partir de ce modèle intermédiaire.

Dans la dernière partie qui fait office de conclusion nous passons en revue les travaux précédents sur le sujet et nous discutons ensuite l'efficacité de cette solution (qui a été mise en oeuvre avec le système EUCLID).

On trouvera en annexe quelques exemples de solides reconstruits par ces algorithmes.

I Introduction au sujet de recherche

I -1 La C.A.O des solides

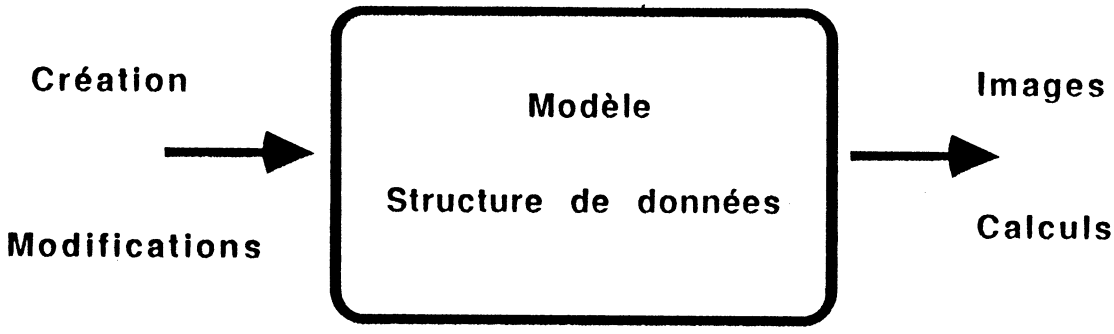
Les systèmes de C.A.O (Conception Assistée par Ordinateur) sont des systèmes informatiques destinés à assister les bureaux d'études industriels dans leurs tâches de conception. Leur emploi se partage approximativement ainsi :

- Construction mécaniques, aéronautique, bâtiment ... 40%
- Electronique 40%
- Autres 20%

Nous nous intéressons ici au premier groupe que l'on appelle C.A.O volumique ou solide, car les objets conçus par ces systèmes sont des assemblages de solides tridimensionnels (pièces mécaniques, bâtiments, etc ...).

Un système de C.A.O est centré sur une représentation des objets dans la mémoire de l'ordinateur, cette représentation est un modèle ou abstraction qui permet de manipuler des solides virtuels sans avoir à les produire réellement. Le système de C.A.O est organisé autour de la représentation (structure de donnée ou base de donnée suivant qu'elle réside en mémoire vive ou sur disque) et propose deux types d'outils : les uns pour enrichir les modèles en créant ou en modifiant des solides, les autres pour extraire de ces modèles les informations utiles à la conception et à la fabrication.

Physiquement le système se compose d'un ordinateur hôte, d'un poste de travail interactif muni d'un écran graphique de bonne définition et en général d'un traceur pour sortir des dessins.



Nous verrons en détail dans la deuxième partie les différentes techniques utilisées pour représenter les solides. Retenons tout de suite qu'il existe deux grandes familles de systèmes:

- Les systèmes incomplets qui mémorisent une information partielle sur le solide : soit des plans en deux dimensions (systèmes de D.A.O Dessin Assisté par Ordinateur), soit des fil-de-fers en trois dimensions. Ces systèmes permettent de sortir des dessins plus facilement qu'à la main mais ils ne vont pas plus loin. Ils sont pourtant très répandus car ils sont plus anciens et peu couteux, particulièrement beaucoup existent sur micro-ordinateurs.

- Les modeleurs solides qui mémorisent un modèle complet du solide, ce sont les systèmes du futur car ils permettent une véritable conception avec l'ordinateur.

Les points cruciaux dans les systèmes de C.A.O sont les suivants :

- Qualité du modèle : c'est la richesse et la précision du système de modélisation des solides. Les meilleurs systèmes permettent de représenter des solides plus complexes et plus précis.

- Efficacité des algorithmes : Les systèmes de C.A.O se distinguent par la robustesse et la rapidité des algorithmes mis en oeuvre. Les systèmes les

plus puissants proposent des fonctions très complexes qui impliquent une programmation avancée. (calculs d'intersections par exemple)

- Facilité d'utilisation : Les systèmes doivent offrir à leurs utilisateurs le plus grand nombre d'outils pour créer et modifier des solides. On considère que la création des solides est un véritable goulot d'étranglement qui consomme une grande partie du temps des utilisateurs.

I-2 Définition du sujet de recherche

Si l'on considère le problème de lire automatiquement des plans de dessin industriel pour le compte d'un système de C.A.O on peut immédiatement le décomposer en sous-problèmes distincts :

-1. Numérisation du plan : C'est-à-dire faire passer l'image du support papier au support informatique. Cette opération se fait en général avec un scanner qui est une barre de cellules photo-électriques qui convertissent l'image en un tableau de points.

- 2. Traitement d'image : L'image sous forme de tableau de points est inexploitable directement. Il faut la traiter pour éliminer les bruits de fond, et la convertir en primitives : segments de droite, de cercles, de courbes. A ce niveau on possède déjà un modèle exploitable par un système de D.A.O bidimensionnel.

- 3. Construction du solide : Il s'agit de passer du modèle bidimensionnel au solide tridimensionnel.

Il existe déjà des systèmes opérationnels pour réaliser les opérations 1. et 2., comme celui que commercialise la société Tektronix. Par contre il n'existe actuellement aucun système commercial qui réalise le troisième point. Des recherches académiques ont été faites sur le sujet, on les trouvera resumées et comparées au présent travail à la fin de ce rapport.

Notre travail se limite en effet à ce troisième point, soit un ensemble de deux ou trois vues bidimensionnelles : reconstituer le ou les solides qui admette ces vues comme projections orthogonales. Nous verrons que nous avons aussi été amenés à résoudre le problème suivant : soit un ensemble de points et de segments tridimensionnels (fil-de-fer), trouver le ou les solides admettant ces points et ces segments comme sommets et arêtes.

I-3 Applications Industrielles

Nous proposons ici un certain nombre d'applications qui montrent que la solution de ce problème peut trouver plusieurs emplois dans un système de C.A.O.

- Récupération de plans :

C'est l'application initiale qui motive notre recherche. Il faut savoir que les plans sur papier représentent encore une majorité écrasante de l'information technique, en particulier dans les petites entreprises qui ne ce sont pas équipées de couteux matériels informatiques. Quand une entreprise s'équipe en C.A.O il lui faudrait reprendre ses anciens travaux mais le coût en est élevé car il faut qu'un opérateur crée manuellement le modèle et nous avons déjà remarqué que cette opération était un facteur limitatif de l'emploi des systèmes de C.A.O. La récupération automatique des plans (ou du moins fortement assistée) représente un défi très important et peut ouvrir un vaste marché.

- Echange de données :

Les systèmes de C.A.O différents échangent des données sur les solides à travers des normes comme I.G.E.S (Initial Graphic Exchange Standart) **[Wilson & all. 82]** ou S.E.T de l'aérospatiale. Dans tous les cas ces standards dégradent la qualité de la représentation, en particulier si le système émetteur du modèle est un système de D.A.O ou un système fil-

de-fer il ne faut pas s'attendre à trouver un modèle solide. Dans ce cas il est très utile de posséder une interface qui sache reconstituer les solides à partir de ces données. Particulièrement on peut imaginer le cas d'une entreprise qui remplace son système de D.A.O par un système de C.A.O solide, elle pourrait alors récupérer automatiquement toutes les anciennes données. Nous avons dit que les plans papier représentaient l'écrasante majorité des archives techniques, il faut ajouter que pour la partie sur support informatique les modèles bidimensionnels ou fil-de-fer représentent encore une écrasante majorité.

- Outil de création de solide :

Nous pouvons encore imaginer d'intégrer des algorithmes de construction de solides à partir de vues bidimensionnelles comme outils interactifs de création de solides. Nous avons déjà répété que la création de solide est un goulet d'étranglement des systèmes de C.A.O. Nous pouvons fournir ici un outil puissant puisqu'il suffit de dessiner des vues du solide pour que le système le reconstruise automatiquement. Remarquons que tous les outils de création de solides sont obligés de faire appel à du dessin en deux dimensions tout simplement parce qu'ils utilisent des écrans graphiques.

C'est dans l'optique de cette utilisation que nous avons particulièrement étudiés des algorithmes qui soient capables de reconstituer un solide à partir de seulement deux vues orthogonales.

I-4 L'analyse de scène

Ce dernier paragraphe cherche à désamorcer une fausse piste, celle de l'analyse de scène. Cette technique est développée par les chercheurs en robotique et en vision intelligente par ordinateur pour analyser les scènes réelles qui peuvent défiler devant les caméras d'un robot. Il s'agit en général d'étudier une vue formée de segments de courbes et d'y reconnaître les différents objets et leur positions dans l'espace. L'idée de

base est d'étudier chaque arête de la scène pour déterminer si elle est convexe ou concave, on peut se reporter à **[Huffman]** ou **[Winston]** pour plus de précision. Cependant ces travaux ne sont pas du tout adaptés à notre problème, en effet le postulat fondamental de ces techniques est que la caméra se trouve dans une position générale par rapport à la scène et en particulier qu'aucune face des objets soit vue de profil. Dans les plans de dessin industriel c'est exactement le contraire qui se produit, les vues sont choisies le long d'axes particuliers c'est le fait qu'il y a plusieurs vues qui permet de reconnaître l'objet.

figure I -1 Une scène de robotique.

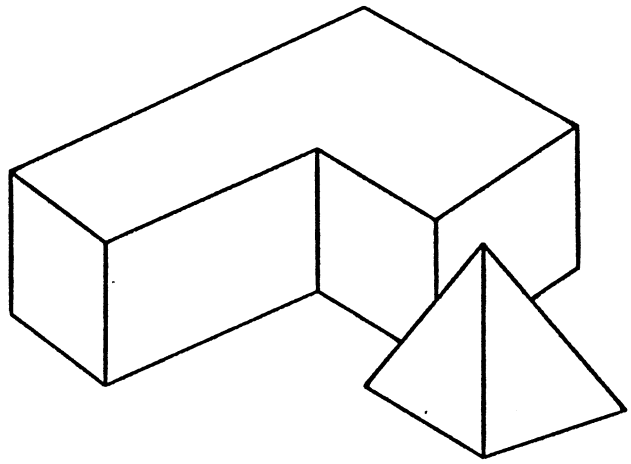
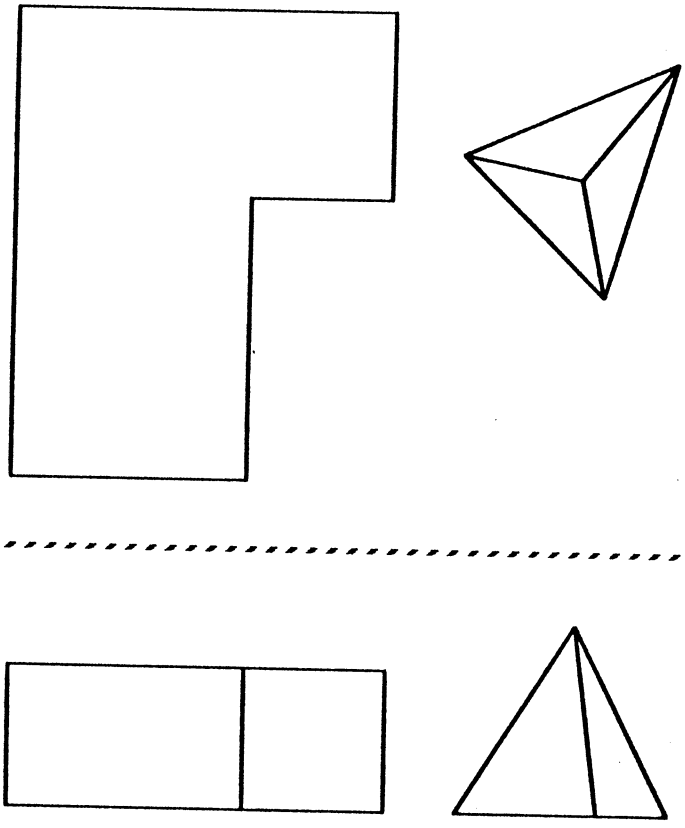


figure I-2 les mêmes objets en vues orthogonales



II - Modélisation des solides

Dans cette deuxième partie nous étudions la modélisation des solides. Pour cela nous commençons par donner une définition des solides et des opérations qui peuvent leur être appliquées. Puis nous étudierons les propriétés que doit présenter un schéma de modélisation des solides. A l'aide de ces définitions nous présenterons ensuite les principaux schémas de modélisation. Nous entrerons dans les détails enfin sur le schéma de représentation par les frontières qui est celui que nous utilisons dans la suite de cette thèse. Une présentation de la modélisation des solides se trouve aussi dans **[Requicha 80], [Requicha & Voelcker 77, 82, 83].**

II-1 Définition des solides

Les définitions suivantes essaient de rendre compte des propriétés intuitives des solides en termes topologiques et géométriques. Un solide est une partie de l'espace Euclidien à trois dimensions E^3 . Toutes les parties de cet espace ne peuvent évidemment être considérées comme des solides, nous allons donner les propriétés caractéristiques des solides. On trouvera des définitions équivalentes dans **[Requicha 80]**.

II-1-1 Rigidité

La forme d'un solide ne dépend pas de son orientation ni de sa position dans l'espace. C'est à dire que si deux parties de l'espace sont l'image l'une de l'autre par un déplacement rigide on peut les considérer comme un même solide. Un déplacement rigide direct est une transformation affine orthogonale directe de l'espace Euclidien. Soit les compositions de rotations et de translations.

II-1-2 Finitude

Un solide a une extension bornée dans l'espace. C'est à dire qu'il existe une sphère de rayon fini contenant entièrement le solide.

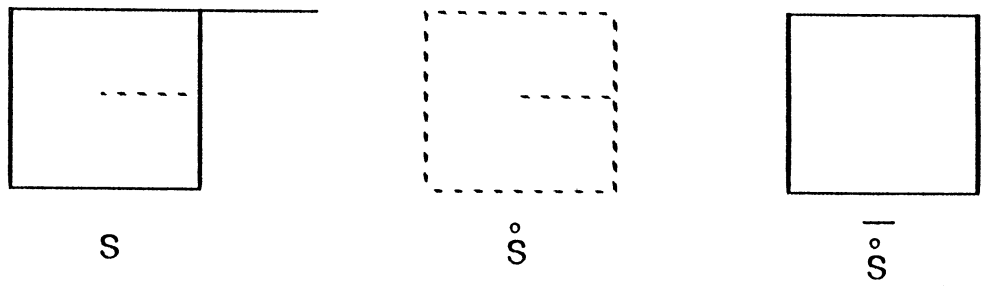
On peut utiliser des régions infinies dans les systèmes de C.A.O, comme les demi-espaces, mais ils faut les considérer comme des intermédiaires et non comme des solides valides.

II-1-3 Homogénéité

Un solide est composé de matière, il doit donc comprendre un "intérieur" et un "extérieur" et pas de parties sans "épaisseur". Ce sont des notions topologiques qui peuvent se formaliser en définissant un ensemble régulier (Requicha parle de r-set) : un ensemble régulier est une partie non vide égale à la fermeture de son intérieur. On appelle régularisation l'opération notée **r(S)** qui consiste à prendre l'intérieur d'un objet S, ce qui supprime toutes les parties inhomogènes, puis à fermer cet intérieur.

figure II-1 Régularisation

$$r(S) = \overline{\overset{\circ}{S}}$$



II-1-4 Opérations Booléennes

Il est souhaitable que nos solides abstraits possèdent l'équivalent des opérations d'usinage sur les solides réels : ajout ou enlèvement de matière. Ces opérations sont réalisées avec les opérateurs ensemblistes ou booléens usuels au prix d'une légère modification. En effet les opérateurs standards ne respectent pas l'homogénéité des solides, c'est pourquoi on a convenu de faire suivre l'opération booléenne par une régularisation et l'on parle d'opérateurs régularisés ou réguliers. Ces opérateurs régularisés sont parfois renommés opérateurs topologiques et on les note avec une étoile pour les distinguer des versions non régularisées .

réunion	$\cup$	régularisée	fusion	$\cup^*$
intersection	$\cap$	régularisée	commun	$\cap^*$
différence	-	régularisée	coupe	$-^*$

figure II-2 Exemple d'intersection régularisée

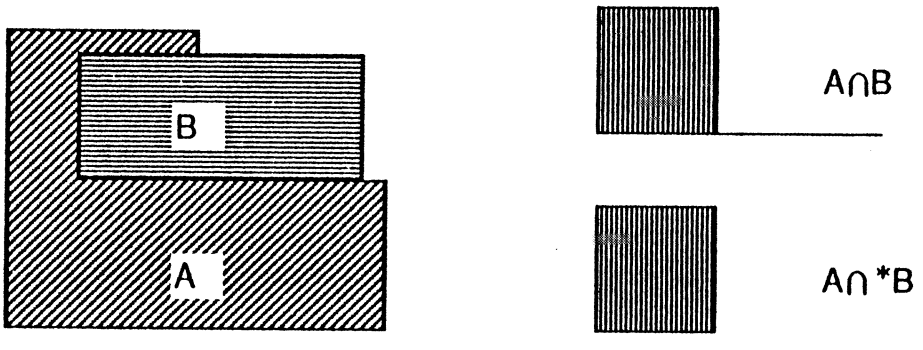
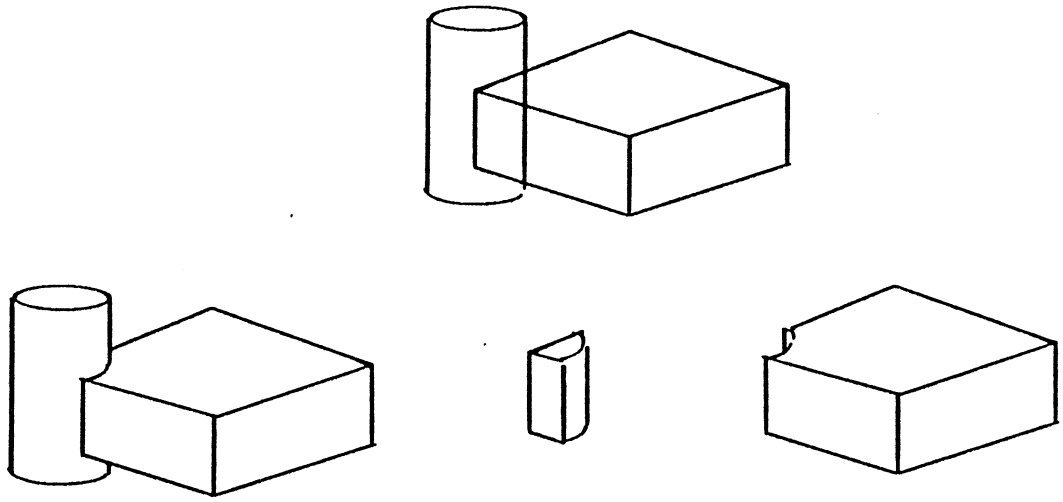


figure II-3 Les opérateurs booléens régularisés



II-1-5 Régularité des surfaces

Pour éviter les cas pathologiques nous devons imposer des conditions sur les surfaces des solides. Les surfaces analytiques couvrent un domaine suffisant (en général les surfaces algébriques suffisent).

Nous appelerons ensemble analytique (resp. algébrique) une partie de l'espace définie par l'équation :

F(x,y,z) $\geq$ **0** où **F** est une fonction analytique (resp. algébrique). Les ensembles semi-analytiques (resp semi-algébrique) sont ceux que l'on peut obtenir à partir des ensembles analytiques (resp. algébriques) et d'un nombre fini d'opérations régularisées : $\cup^*, \cap^*, -^*$.

Dans la pratique industrielle les surfaces utilisées sont des plans, des quadriques ou des surfaces dites "libres" qui sont des surfaces paramétrées polynomiales ou rationnelles (Béziers, B-splines).

II-1-6 Définitions des solides

Nous conviendrons maintenant d'appeler solides les ensembles **semi-analytiques, réguliers** et **bornés**. Cette définition a été proposée par Réquicha **[Réquicha]**.

II-1-7 Opérations sur les solides

Dans un système de C.A.O il est souhaitables de pouvoir effectuer sur les solides plusieurs opérations de modélisation qui permettent de les combiner, de les modifier ou d'en créer de nouveaux.

- Les transformations affines

Rigides ou non rigides. C'est-à-dire : les translations, rotations, symétries, homotéties. Ces opérations de base sont souvent utilisées dans les algorithmes de modélisation il est donc souhaitable que leur implantation soit rapide efficace et robuste.

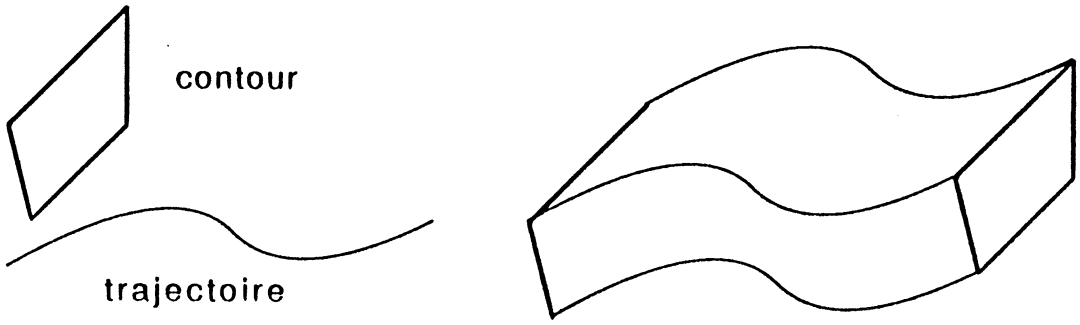
- Les opérateurs booléens régularisés

Ils ont déjà été définis et ce sont de puissants outils de construction, tout système de modélisation des solides doit les proposer et les algorithmes qui les réalisent doivent être efficaces et robustes, en particulier pour les cas particuliers fréquents de surfaces confondues.

- Les balayages

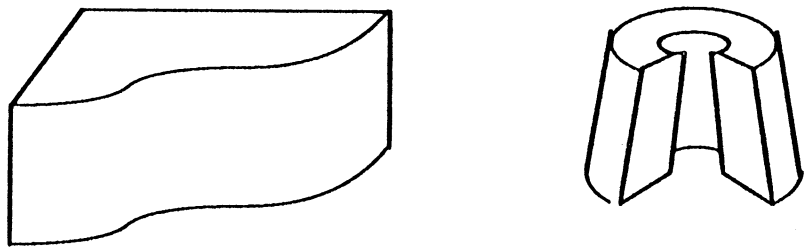
Les primitives de balayage sont largement utilisées en C.A.O pour définir des solides. Un balayage est défini par un contour et une trajectoire : le solide est l'ensemble des points balayés par le contour quand il parcourt la trajectoire.

figure II-5 Balayage



Les balayages ne sont généralement pas implantés dans toute leur généralité mais on doit trouver au minimum les **prismes** ou **extrusions** où la trajectoire est une translation, et les **solides de révolution** où la trajectoire est une rotation.

figure II-6 Balayages simples



prisme solide de révolution

Il va de soi que le système de modélisation doit aussi proposer des outils pour modéliser les contours en deux dimensions, mais la description de tels outils sort de notre propos.

- Les primitives

Tous les systèmes de C.A.O permettent de construire des primitives simples de façon paramétrée. Ces primitives sont en général :

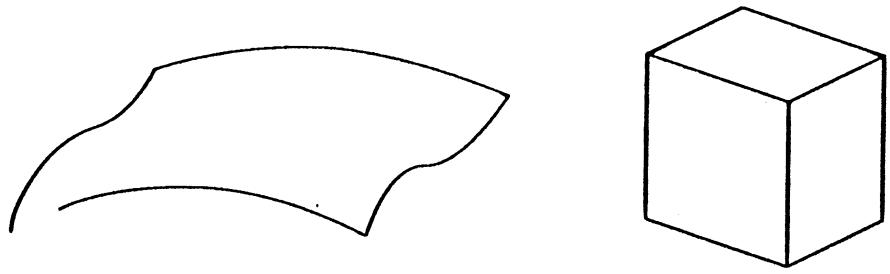
La boite rectangulaire, le cylindre, le cône, la sphère, le tore, éventuellement le coin triangulaire.

Les solides ne comportant que des surfaces simples sont généralement construits à partir de primitives, de balayages simples, de transformations linéaires et d'opération booléennes.

- Les surfaces libres

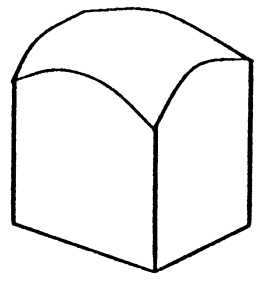
Pour pouvoir utiliser des surfaces libres (surfaces paramétrées) il est généralement nécessaire d'introduire la notion de demi-espace. Un demi-espace est la donnée d'une surface et d'un point et se comporte comme si toute la partie de l'espace délimitée par la surface du coté du point est emplie de matière. Le demi-espace doit être utilisé aussitôt dans une opération booléenne : soit une intersection, soit une coupe où le demi-espace est l'outil de la coupe (ce qui revient à faire un commun avec le demi-espace complémentaire). Dans ce cas on a bien affaire à un solide correct après l'opération booléenne.

figure II-7 Demi-espace



demi-espace

boite



commun

II-2 Composants des solides

Nous allons définir dans ce chapitre les différents éléments géométriques et topologiques qui interviennent dans les solides. Ce sont les sommets, les arêtes et les faces, ainsi que les surfaces et les courbes.

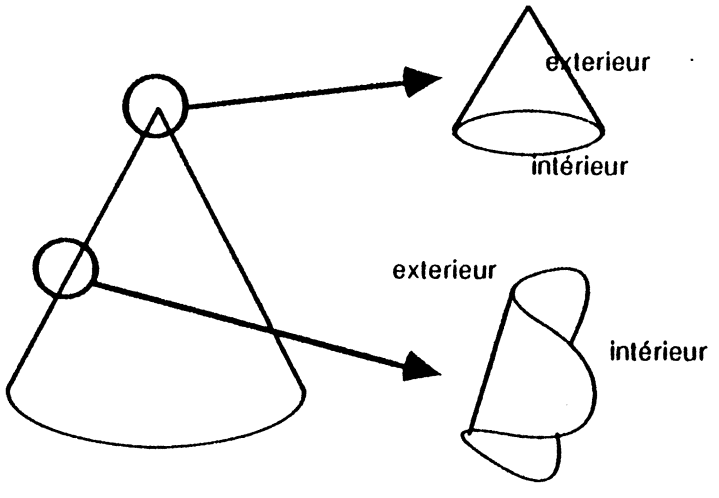
II-2-1 propriétés des surfaces

Nous allons donner ici la définition de quelques propriétés des surfaces. Une surface est une variété bidimensionnelle de E^3. Quand elle intervient dans la frontière d'un solide une surface doit séparer la matière du vide, ce qui implique certaines propriétés. On peut trouver de la géométrie des surfaces dans **[Hilbert]**, **[Faux & Pratt]**.

II-2-1-1 2-manifold

Une surface sera dite 2-manifold si en chaque point de la surface l'intersection d'une petite sphère avec la surface est topologiquement identique à un petit disque (c'est-à-dire par déformation continue). Une surface qui se coupe elle-même n'est pas 2-manifold pas plus qu'un cône complet (avec les deux parties), par contre un demi-cône est 2-manifold. Cette propriété permet de définir localement deux cotés de la surface.

figure II-8 Surface 2-manifold



II-2-1-2 Orientation et genre

Sur une surface 2-manifold nous pouvons définir localement deux côtés de la surface, mais pour pouvoir étendre globalement cette propriété il faut que notre surface soit orientable. C'est-à-dire que nous ne pouvons pas choisir un vecteur normal sur la surface et en le déplaçant continûment le ramener au même point avec une orientation opposée. Le prototype de la surface non-orientable est le ruban de Moëbius bien connu.

La propriété importante des surfaces pour l'étude des immersions des graphes est le **genre**. Un théorème fondamental de topologie montre que toute surface est topologiquement équivalente à l'une des suivantes :

- Surface orientable de genre **p** où **p** est entier :

La surface de genre 0 est la sphère, la surface de genre **p** est la sphère avec **p** poignées ou anses. Le tore est un exemple de surface orientable de genre 1 (il est équivalent à une sphère avec une poignée), on peut aussi présenter la surface de genre **p** comme une surface comprenant **p** trous.

- Surface non orientable de genre **p**.

Les surfaces non orientables sont plus difficiles à se représenter, ce sont des sphères avec des poignées qui traversent la surface reliant l'intérieur à l'extérieur (comme la bouteille de Klein).

On peut représenter les surfaces fondamentales par un circuit formé d'arcs que l'on identifie deux à deux pour former des arêtes. On construit la surface en découpant le circuit dans une feuille et en collant bord à bord les arcs qui forment la même arête.

- Les surfaces orientables sont représentées par les circuits suivants :

aa pour la sphère.

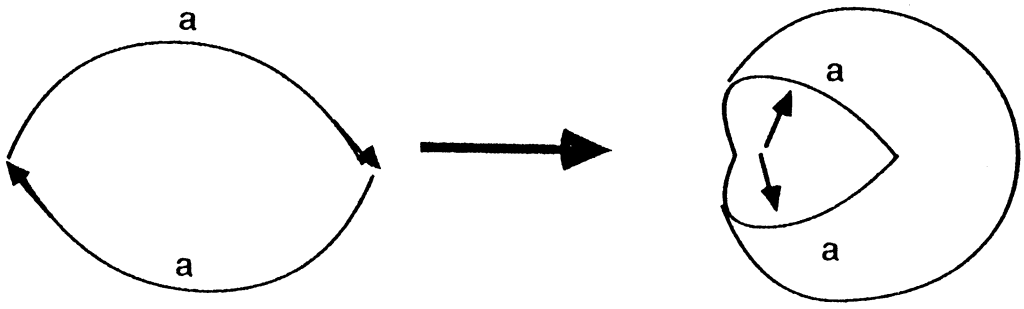
$a_1 b_1 a_1 b_1 \ldots a_p b_p a_p b_p$ pour la surface de genre **p**.

- Les surfaces non orientables sont représentées par les circuits

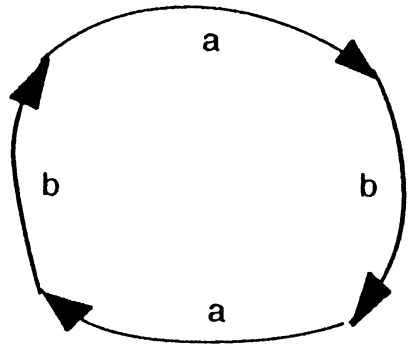
$a_1 a_1^{-1} a_2 a^{-1} \ldots a_p a_p^{-1}$

figure II-9

Sphère



tore



Nous ne nous intéresserons évidement qu'aux immersions de graphes dans des surfaces orientables pour pouvoir avoir des solides. On voit que puisque toutes les surfaces de même genre sont topologiquement équivalentes, pour savoir si un graphe est immergeable dans une surface il suffit de considérer le genre de la surface. A partir d'une immersion d'un graphe dans une surface on peut en donner une dans une autre surface de même genre par déformation continue.

II-2-1-3 Régularité

Une surface régulière est une surface où il est possible de définir en chaque point un plan tangent. En fait nous utiliserons des surfaces pseudo-régulières c'est-à-dire qu'elles seront régulières sauf en des

points isolés (sommet de cône par exemple) ou le long de certaines courbes. Ces courbes sont le lieu de discontinuités de premier ordre de la surface (discontinuité du plan tangent). Le plan tangent est associé à un vecteur normal, il y a deux possibilités d'orientation de ce vecteur normal le long de la perpendiculaire au plan tangent, nous associerons le vecteur normal à la position de la matière par rapport à la surface : **la direction pointée par le vecteur normal est l'extérieur du solide.**

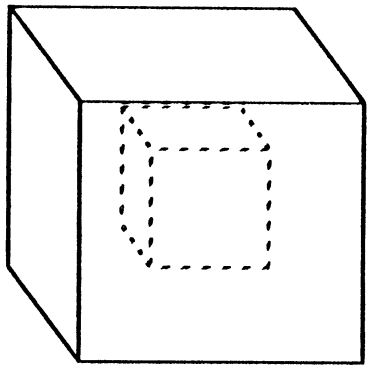
II-2-1-4 Ouverture

Une surface peut être ouverte ou fermée : les surfaces ouvertes ont une extension infinie ou un bord, les surfaces fermées n'ont pas de bord et leur extension est limitée dans l'espace. Le plan est un exemple de surface ouverte, la sphère de surface fermée.

II-2-2 Faces, arêtes et sommets

Nous pouvons considérer toute la frontière d'un solide comme un ensemble de surfaces fermée. Quand il y a plusieurs surfaces cela signifie que le solide est déconnecté ou qu'il contient des cavités. Chacune de ces surfaces sera appelée une **nappe** du solide.

figure II-10 nappes



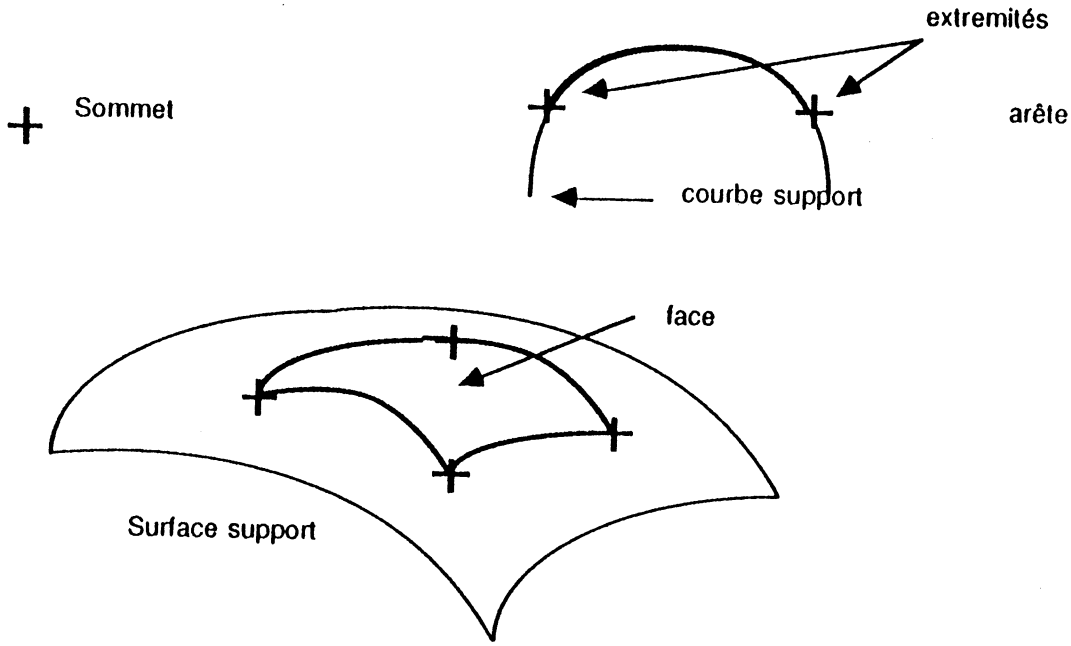
Cube avec cavité :

——— nappe extérieure

- - - - - nappe intérieure

Une nappe est composée de plusieurs morceaux de surfaces réguliers séparés par des courbes de points de discontinuités. Considérons l'ensemble des points réguliers d'une nappe, chaque composante connexe de cet ensemble est une **face**. Cette définition recouvre la notion traditionnelle de face. Les **arêtes** sont les segments de courbe formés par les points irréguliers. Les **sommets** sont les points irréguliers isolés ou ceux par lesquels passent plusieurs arêtes. Une arête est un segment de courbe limité par deux sommets, une face est une portion de surface limitée par des arêtes appartenant à la surface. Remarquons qu'une face peut faire partie d'une surface irrégulière, non orientable et non 2-manifold tant que ces défauts ne se trouvent pas dans la partie de la surface à l'intérieur de la face.

figure II-11 sommets, arêtes, faces

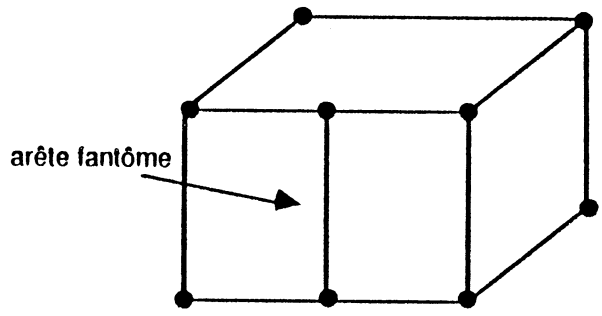


II-2-3 Fil-de-fer, arêtes fantômes, tangence et silhouettes

On appelle **Fil-de-fer** (en anglais **Wireframe**) associé à un solide l'ensemble formé des arêtes et des sommets du solide. C'est cet ensemble qui est utilisé pour représenter le solide dans les systèmes de C.A.O primitifs du type fil-de-fer.

Si deux faces d'un solide se trouvent sur la même surface, avec la même orientation et partagent une arête alors cette arête n'a pas lieu d'exister et les deux faces peuvent être rassemblées en une seule. De telles arêtes que nous baptisons **arêtes fantômes** ne font pas strictement partie du fil-de-fer mais peuvent s'y trouver.

figure II-12 Arête fantôme



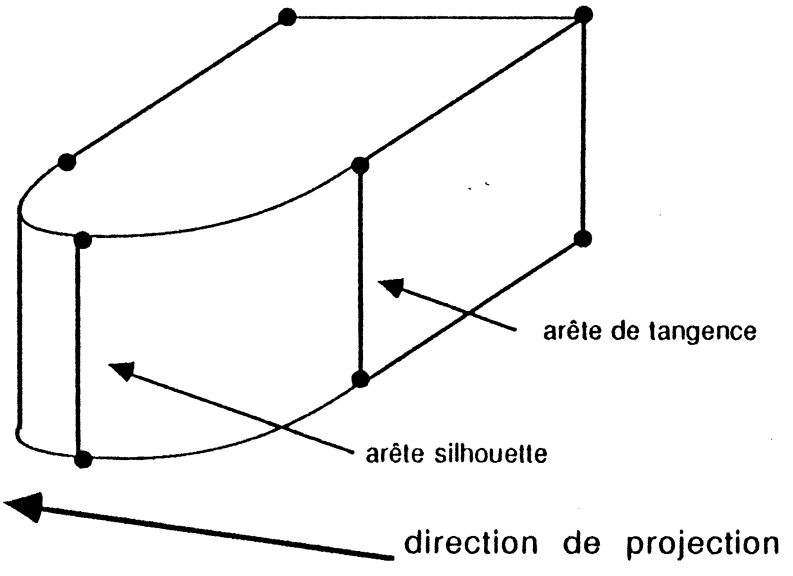
Une **arête de tangence** est une arête séparant deux faces telles que les deux surfaces aient un raccord du premier ordre (continuité du vecteur normal) le long de l'arête. De telles arêtes sont parfois omises dans les dessins industriels, mais nous considérons comme incorrect un fil-de-fer sans les arêtes de tangence.

La notion d'**arête silhouette** est attachée à une prise de vue du solide. Imaginons que notre solide soit observé (pour faire un dessin par exemple) les rayons lumineux sont parallèles entre eux (vues orthogonales) ou convergent en un point (vue perspective). L'arête silhouette est une courbe tracée sur une surface, lieu des points où la

normale à la surface est perpendiculaire aux rayons lumineux. Ces arêtes peuvent partager une face en morceaux, ce sont donc aussi des arêtes fantômes.

Sur les vues d'un solide les arêtes silhouettes correspondent aux **contours apparents**.

figure II-13 Arête de tangence, arête silhouette



II-3 Techniques de modélisation

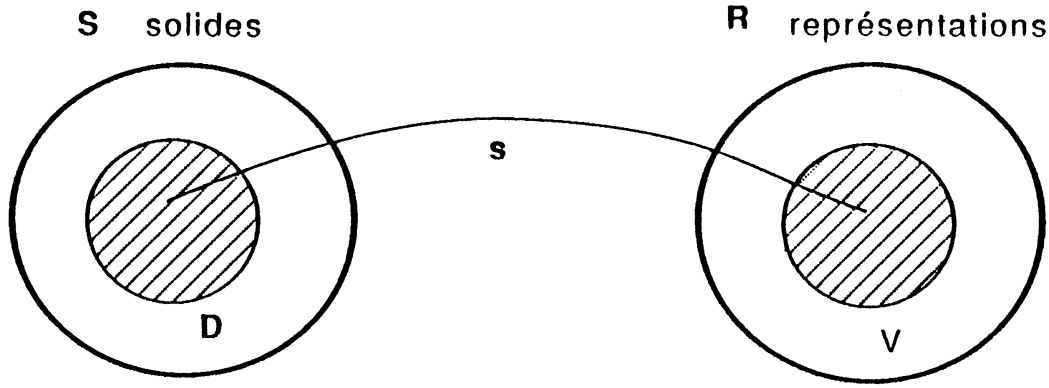
Nous allons maintenant présenter les différentes techniques utilisées pour modéliser des solides. Nous allons commencer par quelques définitions pour pouvoir cerner les qualités et les défauts que l'on peut attendre dans un schéma de modélisation.

II-3-1 Propriétés des schémas de modélisation

Un schéma de modélisation des solides est un langage au sens informatique, c'est à dire un ensemble de constructions symboliques à partir d'un alphabet en suivant les règles d'une **grammaire**. Cette

grammaire définit l'ensemble des constructions syntaxiquement correctes du langage, cette correction peut facilement être maintenue par des algorithmes. La **sémantique** du langage définit les solides qui correspondent aux constructions syntaxiquement correctes. Nous appelerons **S** l'ensemble des solides réguliers que nous avons définis précédemment et que nous cherchons à modéliser, et nous appelerons **R** l'espace des constructions syntaxiquement correctes du langage de modélisation. Nous noterons enfin **s** la relation entre **S** et **R** qui lie solides et représentations. Il arrive que **s** ne soit pas définie pour tout les éléments de **S** et **R**, nous noterons **D** le domaine de **s**, c'est-à-dire le sous-ensemble de **S** des solides qui peuvent être effectivement modélisés par **S**. De même nous noterons **V** le sous-ensemble de **R** des représentations qui correspondent effectivement à des solides. Toute ces notations sont reprises sur le schéma suivants.

figure II-14



A partir de ce modèle nous pouvons donner quelques propriétés des schémas de représentations.

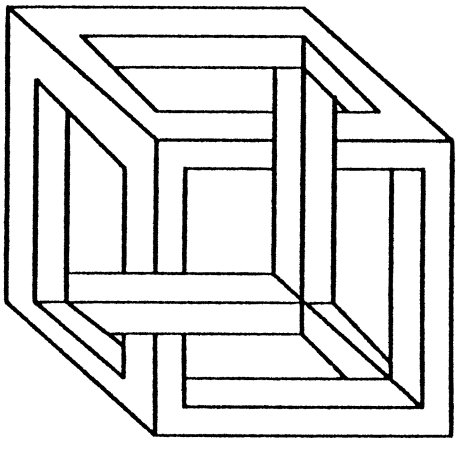
II-3-1-1 Domaine

Le domaine D doit être suffisamment étendu pour que le schéma soit intéressant. L'idéal serait de recouvrir l'ensemble des solides S. L'étendue de D mesure le pouvoir descriptif du schéma.

II-3-1-2 Validité

Un schéma valide est un schéma pour lequel les représentations de tous les modèles de **V** sont bien des solides, c'est à dire que **D** est bien inclus dans **S**. Pour illustrer cette notion , supposons que l'on représente les solides par des vues en perspective, on sait qu'alors il existe des dessins qui on l'apparence correcte (localement) mais qui ne correspondent à aucun objet.

figure II-15 une vue invalide



Nous verrons plus tard que cette propriété n'est pas toujours facile à assurer (par exemple nous avons vu que les demi-espaces pouvaient introduire des solides invalides si l'on ne précisait pas exactement leurs règles d'utilisation). Sur certains modèles la validité des objets doit être confirmée par un utilisateur interactif. Ceci réduit fortement les capacités du système et doit être prohibé. Il faut soit assurer que toutes les

constructions sont valides, soit fournir des algorithmes qui vérifient la validité d'une construction.

II-3-1-3 Complétude

La propriété de complétude exige qu'il n'y ait pas de représentation ambiguë, c'est-à-dire une construction syntaxiquement correcte pour laquelle existe deux solides. Cette propriété est absolument indispensable pour faire de la modélisation solide. Nous verrons plus loin que certains systèmes de modélisation sont ambigus et donc inutilisables pour les applications avancées de C.A.O.

II-3-1-4 Unicité

L'unicité exige que chaque solide n'ait qu'une seule représentation. En fait cette propriété est très rarement présente du fait des permutations toujours possibles dans les structures. Cependant il est intéressant de disposer d'algorithmes qui permettent de tester rapidement l'égalité de deux représentations (c'est à dire qu'elles correspondent au même solide).

II-3-1-5 Propriétés informelles

Certaines propriétés importantes sont plus difficiles à formaliser mais interviennent dans l'évaluation d'un schéma de représentation. Ce sont :

- La concision des représentation.

- La facilité d'emploi : on préferre des schémas qui soient proches des représentations intuitives des solides chez le concepteur.

- L'adéquation à l'usage : Certains schémas comme nous allons le voir s'adaptent mieux à certaines applications. On est d'ailleurs souvent conduit à adopter des multi-représentations, c'est à dire à associer

plusieurs modèles d'un solide. Il faut alors se préoccuper aussi des problèmes de cohérence entre les différentes représentations d'un même solide.

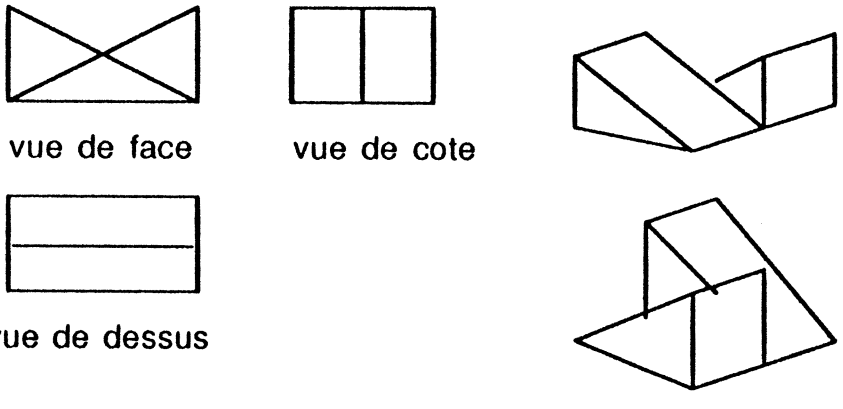
II-3-2 Représentations ambiguës

Les premiers schémas utilisés pour représenter les solides, avant et avec les premiers programmes de C.A.O, sont les plans de dessin industriel et les modèles fil-de-fer.

II-3-2-1 Projections

Un ensemble de vues bidimensionelles ne représente un solide que dans l'esprit du dessinateur, pas pour l'ordinateur. C'est pourquoi ce type de modèle a peu d'utilité hors de reproduire rapidement des dessins. Aucun calcul ne peut etre effectué et il est même difficile de maintenir la cohérence entre les vues. Un exemple très simple montre que ce schéma est ambigu, les dessinateurs ajoutent parfois d'autres données pour lever les ambiguïtés : coupes, vues axonométriques, etc ...

figure II-16 projections orthogonales ambiguës



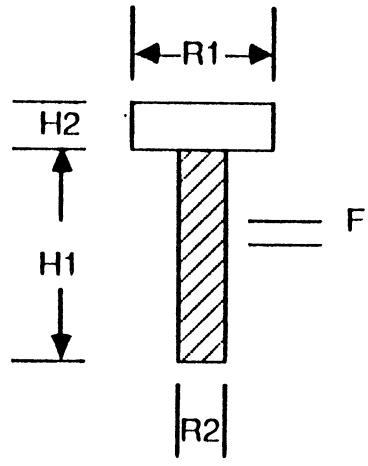
II-3-2-2 Fil-de-fer

Le fil-de-fer est le premier schéma de représentation qui soit proprement informatique. On ne peut hélas l'utiliser que pour faire des dessins sans élimination de parties cachées (mais dans n'importe quelle direction, ce qui est déjà un progrès par rapport au pur schéma bidimensionnel). Le fil-de-fer ne contient en général pas assez d'informations pour représenter un solide de façon non ambiguë comme le montre la figure suivante. Nous étudierons dans la troisième partie sous quelles conditions un fil-de-fer peut représenter un solide unique.

II-3-3 Instanciations de primitives

Certaines applications spécifiques représentent les solides par des instanciations de primitives paramétrées. Le système possède alors un domaine très réduit, par exemple l'ensemble des vis de la figure définies par cinq paramètres.

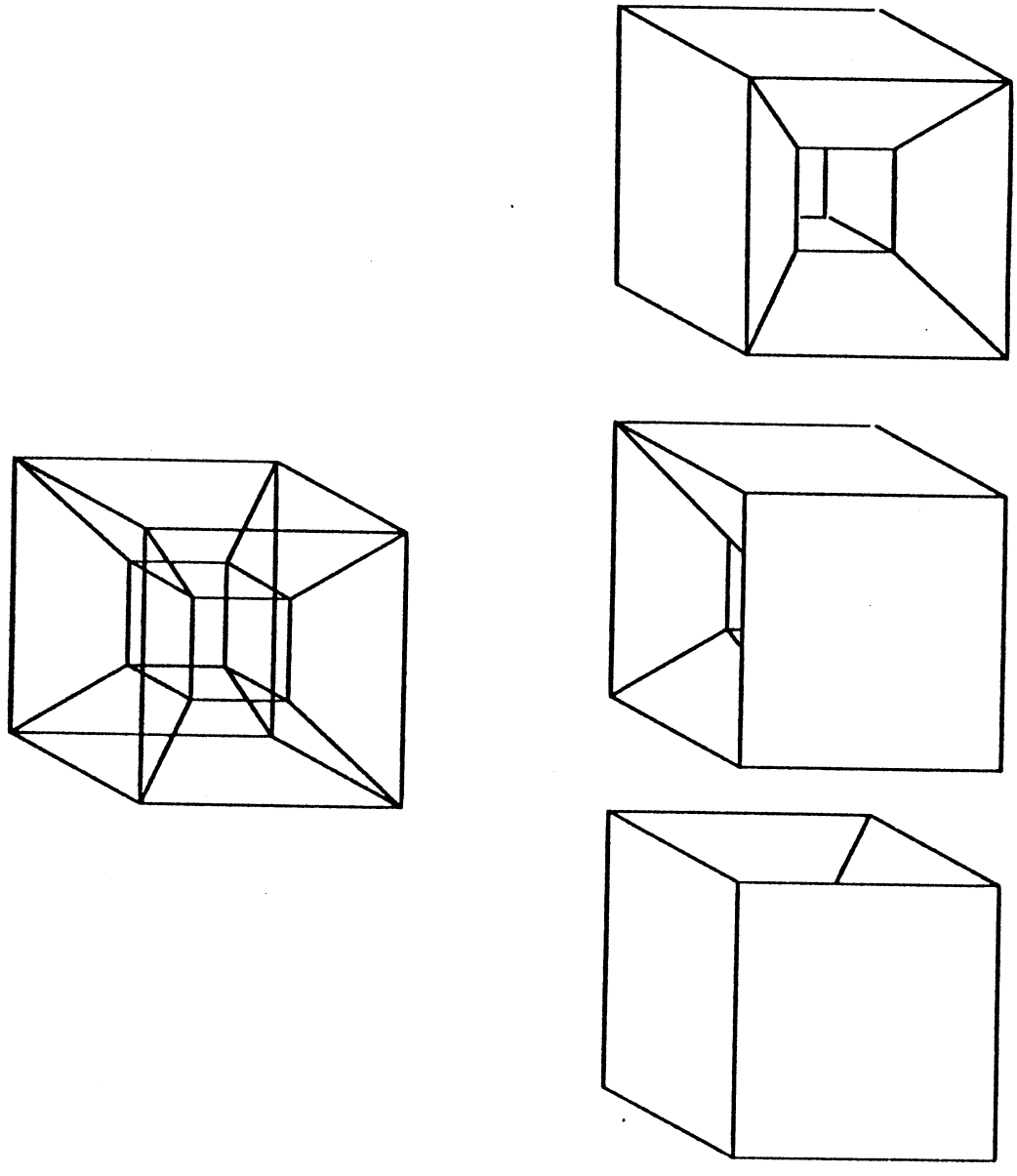
figure II-18 pièce paramètrée



La pauvreté de ce modèle le réduit à des applications comme la technologie de groupe. Par contre il présente bien des avantages du point de vue de la validité et de l'unicité des descriptions. Un autre

inconvénient et qu'il faut associer des algorithmes pour chaque classe générique. Notons qu'il est recommandé dans tout système de C.A.O de fournir des outils pour créer de telles familles de pièces paramètrées.

figure II-17 Un fil-de-fer ambigu avec ses trois interprétations solides



II-3-4 Décomposition spatiale

On regroupe sous ce nom une série de techniques qui reposent sur la décomposition d'un solide en petits sous-solides élémentaires simples (cubes, tétraèdres, ...), ces techniques sont en général employées dans des applications particulières, leurs principaux défauts étant le grand volume de données généré, la difficulté de faire des images, l'approximation imparfaite des solides qui dépend du degré de résolution. En général ces représentations sont déduites à partir d'autre représentations pour un usage particulier. On peut distinguer :

- Les tableaux spatiaux

Cette technique découpe l'espace en éléments réguliers (cubes) appelés **voxels** (par analogie avec pixels). Le solide est décrit par la liste des voxels occupés. Cette technique peut être utilisée quand les solides sont de forme particulière (bâtiments) ou quand la précision est faible (tomographie médicale).

- Les décompositions

La décompositions en éléments simples (tétraèdres, héxaèdres) est une variante du modèle précédent où la position et la taille des éléments n'est pas fixée par une grille. Ce modèle est très étudié car c'est celui des maillages pour les calculs d'éléments finis. De nombreuses recherches sont faites sur la construction de ces modèles, en particulier pour des solides présentant des surfaces courbes.

- Les octrees

Les octrees sont une généralisation astucieuse des voxels qui consiste à représenter un objet par un cube qui peut prendre trois états :

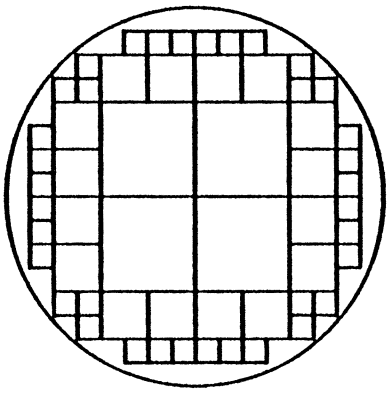
- plein : le cube est emplit de matière.

- vide : le cube ne contient pas de matière.

- mixte : le cube contient du vide et de la matière, il est alors récursivement découpé en huit sous-octrees.

Les cubes de dimension inférieure à une certaine résolution ne sont pas décomposés. L'équivalent à deux dimensions des octrees sont les quadtrees. On voit sur la figure suivante la représentation d'un disque par un quadtree (seuls les éléments pleins sont représentés). Ce modèle a suscité de grands espoirs surtout avec la conception de circuits intégrés spécialement conçus pour traiter les octrees. Mais il présente le défaut de devenir trop volumineux quand on veut une approximation raisonnable des surfaces. Il est cependant utilisé en annexe d'autre représentations pour certains calculs, en particulier pour générer des décompositions en éléments finis. Voir **[Jackins & Tanimoto 80], [Meagher 82]**.

figure II-18 Un disque en quadtree

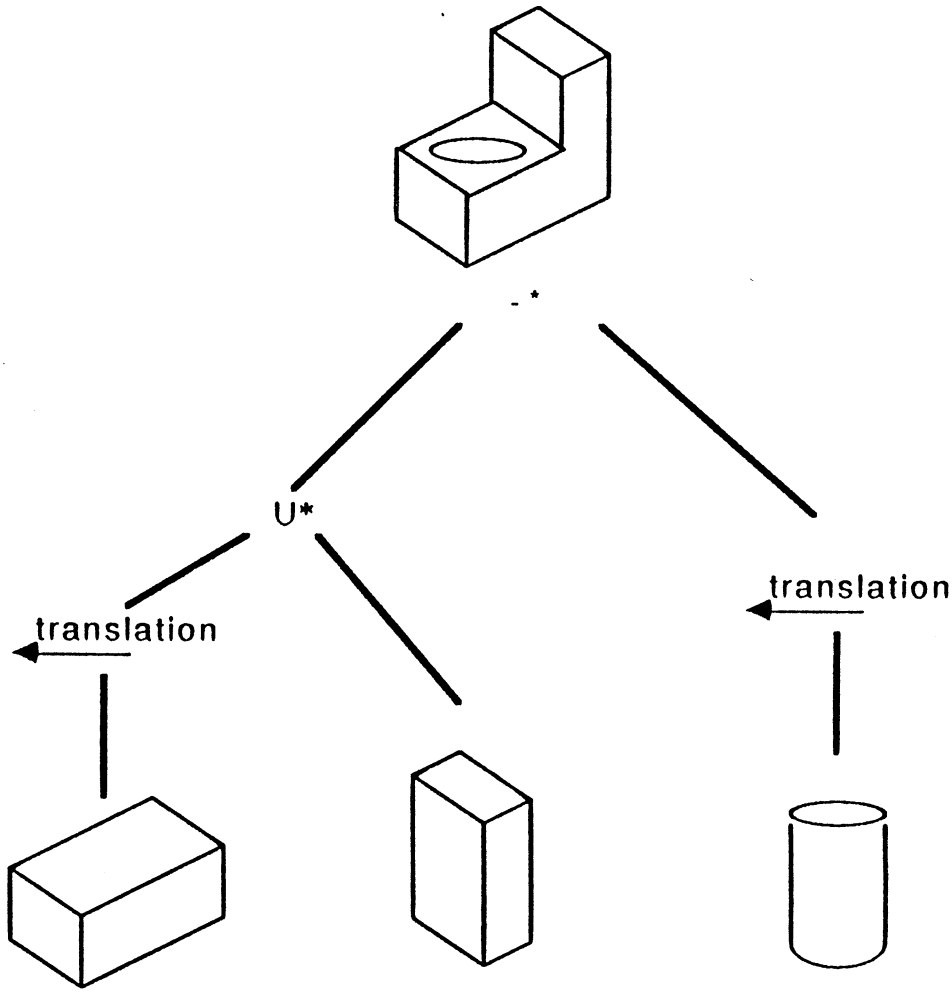


II-3-5 Géométrie constructive

La géométrie constructive (en anglais **CSG** pour Constructive Solid Géometry) est une technique universelle et puissante pour représenter les solides, c'est en fait une des deux techniques utilisées dans les systèmes de C.A.O avec la représentation par les frontières que nous détaillerons après. L'idée de la géométrie constructive est de représenter

les solides par des arbres dont les feuilles sont des primitives simples, et dont les noeuds sont des déplacements ou des opérateurs booléens régularisés. La figure suivante montre un solide représenté en géométrie constructive.

figure II-19 géométrie constructive



Avantages de la géométrie constructive

Les solides construits en géométrie constructive sont toujours valides car ils sont construits à partir de primitives valides et d'opérations qui

préservent la validité. Le domaine couvert par la géométrie constructive est très large puisqu'il s'agit de tous les solides ayant les mêmes surfaces que les primitives. Les représentations sont concises, proches de l'intuition et faciles à construire, certains font remarquer que cela vient du fait que l'on enregistre les opérations à effectuer en retardant les véritables calculs. Cependant il existe des algorithmes qui fonctionnent directement sur la représentation de la géométrie constructive, ce sont les algorithmes de lancer de rayons qui calculent l'intersection d'une droite (rayon) avec un solide. Ces algorithmes utilisent un schéma récursif, c'est-à-dire qu'ils savent calculer avec une primitive et connaissant le résultat sur des solides ils savent calculer pour une combinaison de ces solides. Grâce à ces algorithmes on peut effectuer certains calculs directement (images, volumes) voir **[Roth]**, **[Tilove]**.

Inconvénients de la géométrie constructive

La géométrie constructive n'est pas un modèle **évalué**, c'est à dire que certaines données du solide ne sont pas calculées comme les courbes d'intersections entre surfaces (arêtes) or ces données sont indispensables pour certains algorithmes (dessin précis, commande numérique) on est donc contraint en général d'associer au schéma géométrie constructive une représentation frontière qui est évaluée par un algorithme assez couteux. **[Requicha & Voelcker]**, **[Tilove]**.

L'espace de modélisation de la géométrie constructive pure ne comprend pas les surfaces libres, on peut y remédier en ajoutant la primitive de demi-espace comme on l'a vu, mais il faut alors restreindre son usage pour préserver la validité.

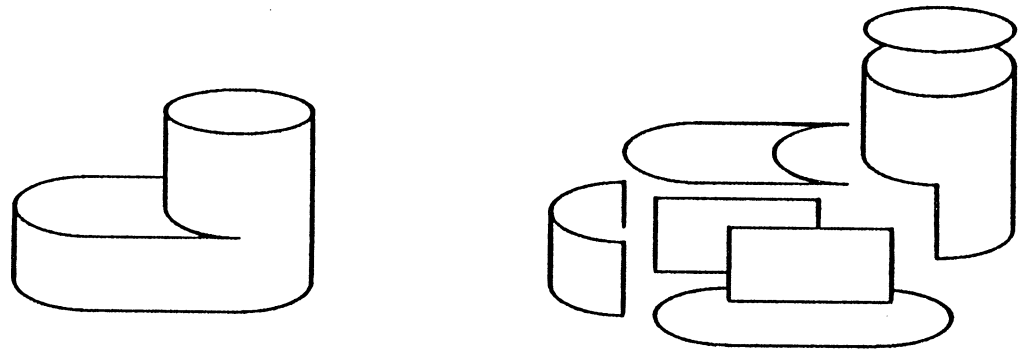
Il est aussi difficile de représenter les balayages (prismes, révolutions) qui sont cependant largement utilisés à cause de leur analogie avec les opérations d'usinage (fraise, tour). Deux possibilités se présentent : soit ajouter les balayages comme primitives ce qui complique sérieusement

par rapport aux primitives traditionnelles (boites, cylindre, cône, sphère ...), soit fournir des algorithmes qui convertissent les balayages en arbres CSG, ce qui peut générer des arbres volumineux et peu intuitifs.

II-3-6 Représentation par les frontières

La représentation par les frontières (en anglais Boundary Representation ou B-Rep) représente un solide par l'ensemble des faces qui constituent sa frontière.

figure II-20 solide limité par sept faces : 5 planes et 2 cylindriques.



Les représentations frontières sont les plus répandues car elles ont été utilisées avant la C.A.O par l'infographie pour faire des images en trois dimensions avec élimination des parties cachées. De plus c'est le seul modèle à partir duquel il est possible de tout calculer. Nous allons beaucoup parler de la représentation frontière car c'est celle que nous utilisons dans notre travail. Ce choix vient de ce que les données sur lesquelles nous travaillons (plans, fil-de-fers) sont plus proche de cette représentation qui se prête plus facilement à une construction automatique, le modèle C.S.G lui se prête mieux à la construction interactive.

Un cas particulier important de représentation frontière est la **représentation polyédrique** qui approxime le solide par un polyèdre à faces planes, on parle souvent de **facettes** quand il s'agit de petites faces qui approximent une surface courbe. De nombreux systèmes utilisent cette représentation pour améliorer les performances des algorithmes, associée à une représentation précise qui permet de régénérer le polyèdre avec n'importe quelle précision.

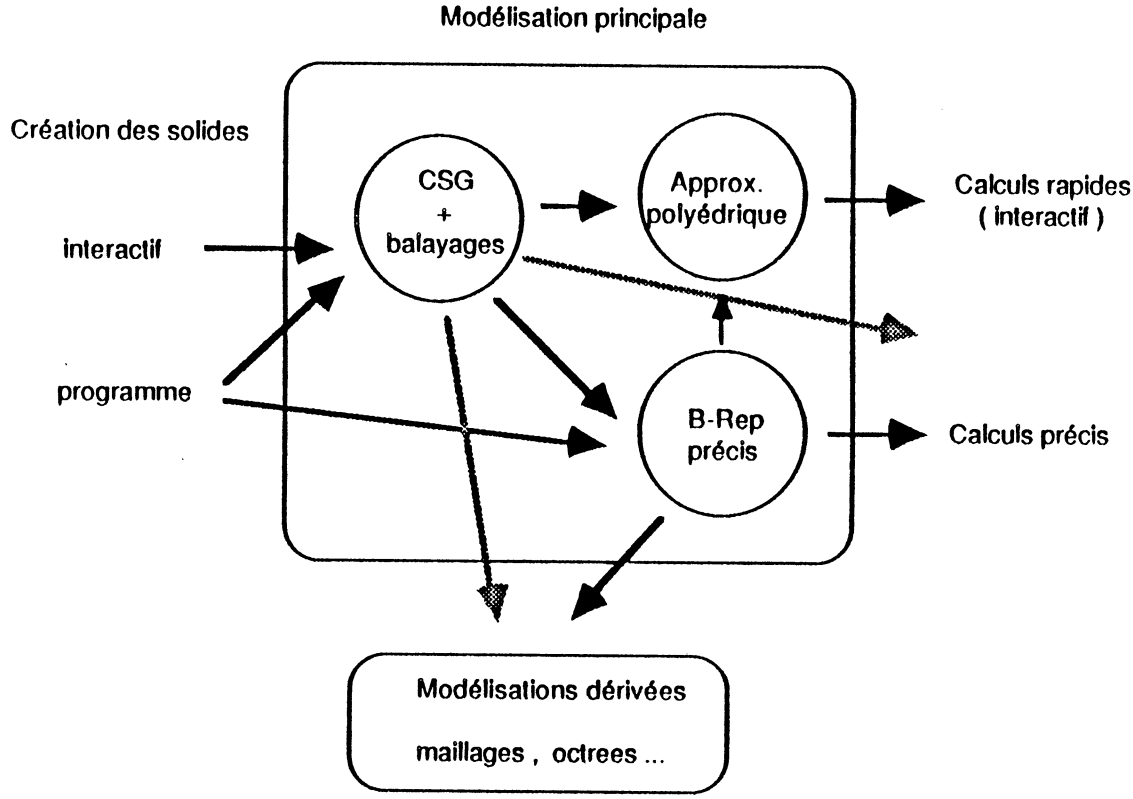
Par analogie avec les polyèdres à faces planes nous parlerons de **polyèdres topologiques** pour les solides représentés par leur frontière.

La représentation frontière est celle qui contient le plus d'informations évaluées sur le solide elle se prête donc au plus grand nombre d'applications. Par contre elle est plus volumineuse et moins intuitive que la géométrie constructive. Le principal problème est celui de la validité . Nous allons examiner en détail cette représentation dans un chapitre suivant.

II-3-7 Multi-représentation

La plupart des systèmes de C.A.O contemporains (PADL-2, Geomod, EUCLID, CATIA) utilisent des multireprésentations. C'est-à-dire que plusieurs schémas de représentation sont utilisés pour un même objet . En général l'architecture d'un modeleur solide est celle représentée sur la figure suivante :

figure II-21 multi-représentation



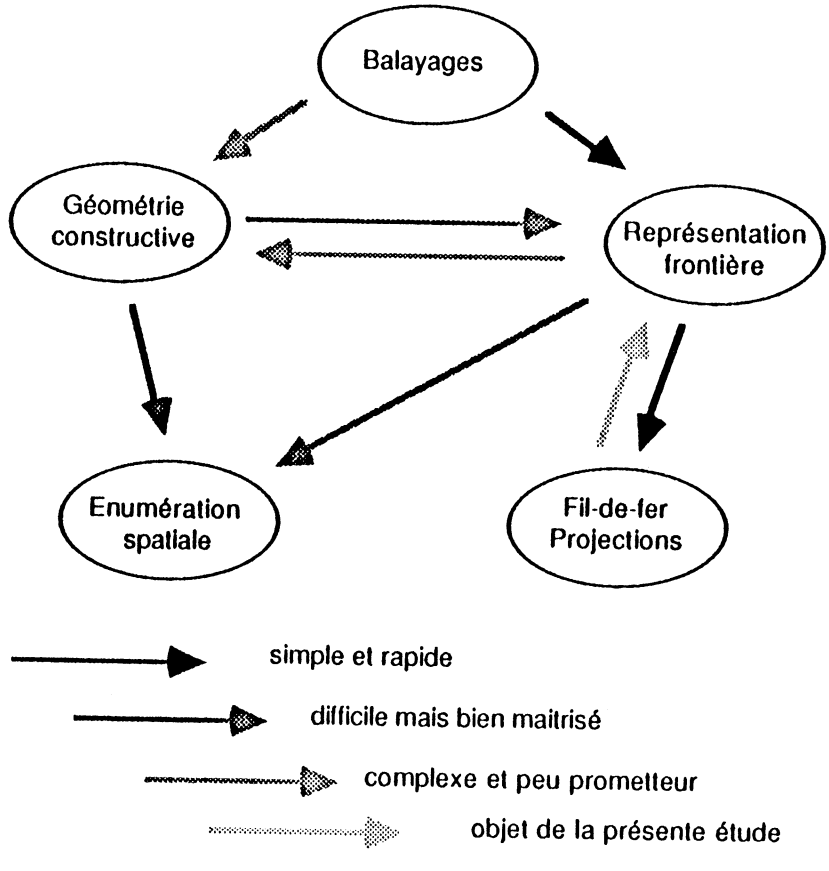
La géométrie constructive est utilisée pour la construction des solides et souvent aussi pour leur stockage en base de données, c'est le modèle de réference. Tous les systèmes cependant ne conservent pas l'arbre de construction. L'avantage de le conserver est de disposer de possibilités d'édition pratiques.

La représentation frontière est utilisée pour ses performances, et plus encore l'approximation polyédrique. En effet certains calculs (calcul de volumes, opérations booléennes, parties cachées) ne s'effectuent de façon satisfaisante dans certains cas que sur cette approximation polyédrique. Cependant peu de systèmes se contentent du polyèdre qui est imprécis. Il est en général dérivé d'une représentation précise sous le contrôle d'un paramètre de précision qui est peu élevé en utilisation

interactive pour avoir des calculs rapides mais peut être très élevé pour obtenir des résultats précis en utilisation différée.

Nous présentons maintenant un schéma montrant l'état de l'art en matière de conversion de modèles, ce problème se pose à l'intérieur des systèmes à multi-représentation mais aussi pour les transferts d'information entre systèmes différents.

figure II-21 Conversions entre modèles



Nous avons qualifiés de " complexes et peu prometteurs " les algorithmes de conversions prenant pour cible la géométrie constructive. En effet l'aspect très " haut niveau " de la géométrie constructive implique que ces algorithmes doivent avoir un comportement " intelligence artificielle " pour retrouver les primitives pertinentes qui composent un

solide. On sait que ce type de reconnaissance de formes n'est pas le point fort des ordinateurs actuels, en général les résultats obtenus sont assez peu intuitifs ce qui restreint leur usage. Cependant nous nous interdisons d'être pessimistes car de tels algorithmes présenteraient un grand intérêt .

II-4 Représentation frontière

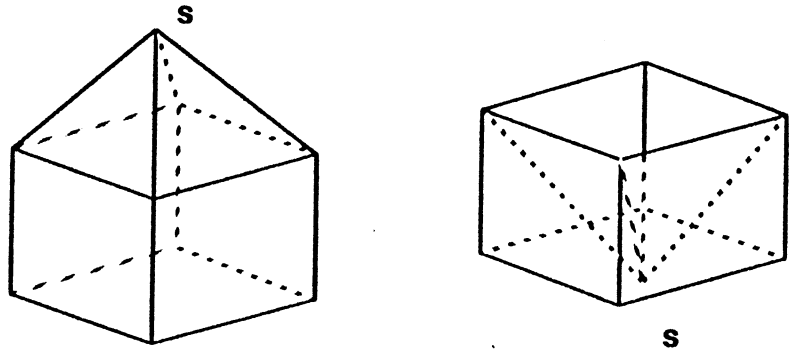
On a l'habitude de distinguer dans une description d'un solide deux types de données :

- Données topologiques : c'est l'ensemble des relations d'appartenance et d'adjacence entre les objets : sommets, arêtes, faces. Les données topologiques définissent une structure associée au solide proche de celle d'un graphe.

- Données géométriques : ce sont les positions dans l'espace des objets qui constituent le solide : coordonnées des sommets, équations des courbes et de surfaces qui portent les arêtes et les faces. Quand on applique une transformation géométrique à un solide on ne modifie que ces données géométriques.

Remarquons cependant qu'il n'est pas possible de découpler complètement topologie et géométrie, par exemple la validité d'un solide ne peut reposer seulement sur sa topologie. Ainsi les deux objets de la figure suivante ont la même topologie mais le second est invalide car certaines arêtes intersectent des surfaces.

figure II-22 topologie correcte Le sommet s est descendu jusqu'à passer en dessous de la face inférieure.



II-4-1 Représentation des surfaces

Nous allons discuter rapidement la modélisation des surfaces en structure de données. Nous pouvons adopter pour ce problème une approche fonctionnelle. C'est-à-dire que nous cherchons d'abord quelles sont les questions géométriques que peut poser un algorithme à une surface pour définir une interface entre l'algorithme et la structure de données. Il est alors possible de représenter des surfaces par des schémas complètement différents du moment que le même jeu de procédures est supporté pour répondre aux interrogations de l'algorithme.

Ce choix vient de ce qu'une représentation unique des surfaces doit être assez complexe pour inclure certains types de surfaces, ce qui oblige à projeter les surfaces les plus simples et les plus courantes dans une structure trop riche qui grève les performances de calcul.

Le jeu de questions formant l'interface avec l'algorithme ne se découvre qu'au cours du développement de ce dernier. Nous allons quand même en énoncer quelques unes. Nous pouvons cependant remarquer que même si nous ne connaissons pas encore toutes les questions, le choix d'une représentation suffisamment "complète" doit permettre de répondre à toutes. Quitte à devoir ajouter plus tard dans la structure de données quelques informations déductibles de celles qui s'y trouvent déjà mais à un coût trop élevé.

Exemples d'interrogations relatives aux surfaces :

- Incidence : un point, une courbe sont-ils sur une surface ?

- Normale : donner le vecteur normal à la surface en un point.

- Intersection : calculer l'intersection de deux surfaces, d'une courbe et d'une surface.

Trois méthodes peuvent être retenues pour représenter une surface, nous allons les décrire en montrant dans chaque cas comment calculer si un point appartient à la surface et comment calculer le vecteur normal.

II-4-1-1 Equation implicite

C'est une équation du type **f(x, y, z) = 0** le signe de la fonction f définit l'orientation naturelle de la surface. Pour savoir si un point est sur la surface il suffit de vérifier l'équation, pour calculer le vecteur normal orienté naturellement il faut différentier :

$$(\partial f / \partial x , \partial f / \partial y , \partial f / \partial z)$$

Ce qui peut se faire par un calcul direct, **f** est souvent une fonction polynomiale. L'exemple le plus simple est l'équation du plan.

$$a\,x + b\,y + c\,z + d = 0$$

Ici le vecteur normal est le vecteur **(a b c)**

II-4-1-2 Equation paramétrique

la surface est représentée par trois fonctions :

$$(x(u,v) , y(u,v) , z(u,v))$$

Nous n'entrerons pas dans la théorie de ces surfaces (Coons, Béziers, B-splines.), voir **[Böhm & all.]**, mais il faut savoir qu'elles sont très utilisées pour modéliser les surfaces industrielles. Ici il est plus difficile de déterminer si un point appartient à la surface en ignorant ses coordonnées paramétriques (u,v) il est cependant possible de projeter le point sur la surface puis de calculer la distance du point à cette projection. La projection se calcule en utilisant une approximation polyédrique de la surface qui donne un point de départ, puis en minimisant la distance **P(u,v) P.**

Quand on connait les coordonnées paramétriques on a facilement accès aux vecteurs directeurs du plan tangent qui sont les dérivées selon u et v et au vecteur normal qui est leur produit vectoriel. Il faut noter que l'interface avec une représentation paramétrique est en général procédurale, c'est à dire que le système fournit une procédure qui permet de calculer à partir des paramètres u,v le point courant et les vecteurs dérivés jusqu'à un certain ordre, cette méthode permet de modifier la représentation de la surface sans changer les algorithmes. Voir **[Faux]**.

II-4-1-3 Représentation PVS

Dans ce travail nous utilisons beaucoup de surfaces élémentaires. C'est-à-dire : le cylindre, le cône, la sphère et le tore. Dans ce cas nous avons utilisé une représentation reposant sur leurs propriétés géométriques. Cette représentation est plus efficace que l'équation implicite **[Goldman], [Wilson 87].** Chacune de ces surfaces est représentée par des paramètres géométriques Points, Vecteurs et Scalaires, on parle de représentation PVS.

- cylindre	axe	rayon du cylindre
- cône	axe	tangente = angle d'ouverture
- sphère	centre	rayon
- tore	cercle	rayon du tuyau

Il est inutile de détailler comment on effectue chaque opération particulière, les calculs se ramènent aux opérations géométriques élémentaires suivantes :

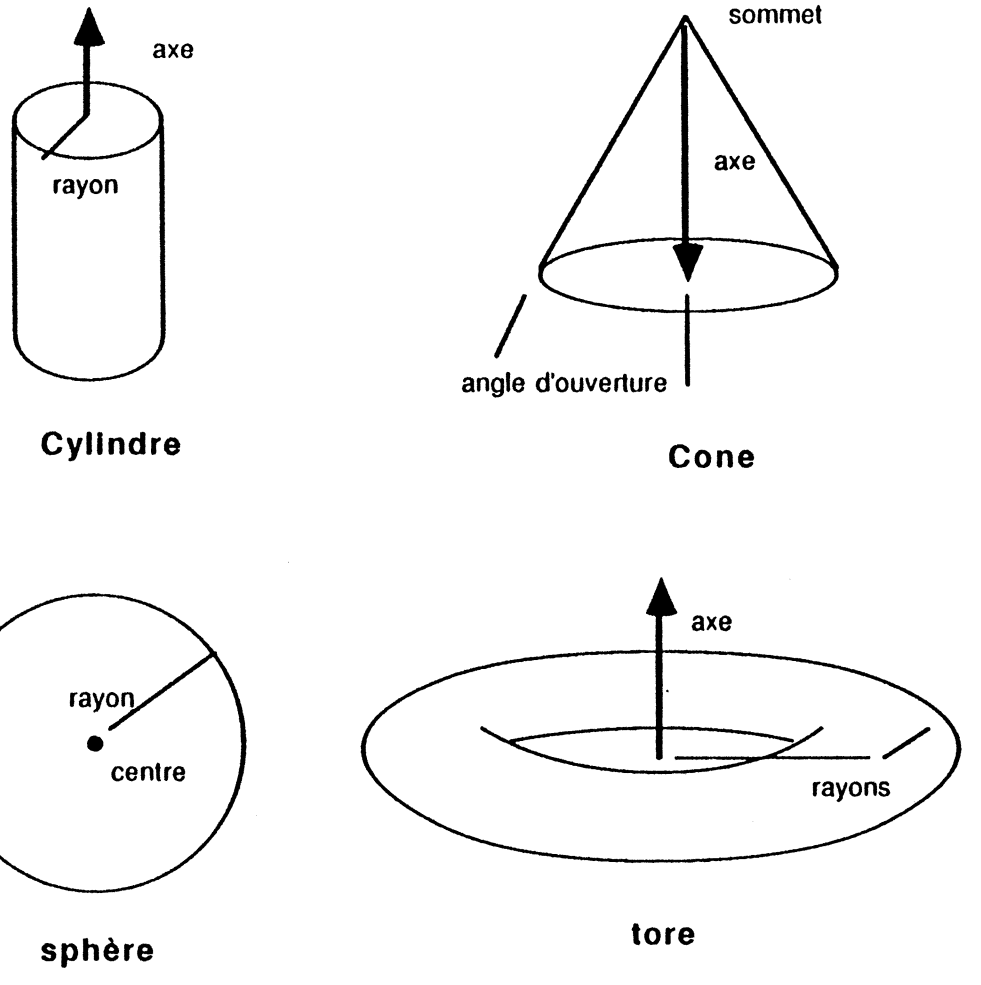
- produit vectoriel, produit scalaire.

- calcul de distance.

- projection d'un point sur un axe ou un cercle.

Il faut par contre comprendre que chaque objet nécessite un calcul simple mais individualisé. Ce modèle n'est pas sans rappeler les langages orientés objets qui fournissent un cadre pratique pour faire fonctionner le mécanisme.

figure II-23 surfaces simples de révolutions



II-4-2 Représentation des courbes

Les courbes sont les objets géométriques qui portent les arêtes.Leur représentation pose les même problèmes que celle des surfaces. Le problème fondamental associé aux courbes et de calculer les courbes

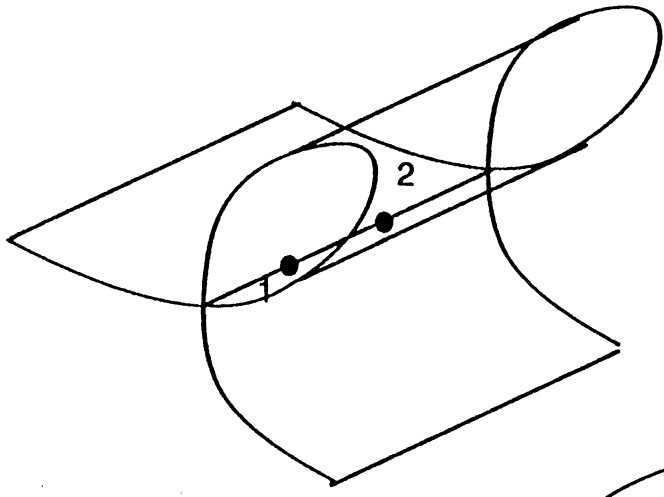
d'intersections qui forment les arêtes à partir des surfaces. Les principales techniques de représentation des courbes sont les suivantes :

II-4-2-1 Intersection de surfaces

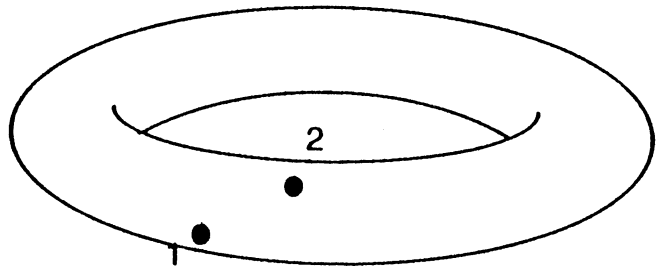
Toutes les courbes que nous allons rencontrer dans un solide sont à l'intersections de deux surfaces qui supportent les faces adjacentes à l'arête associée à la courbe. Une représentation simple de la courbe consiste donc à rassembler deux représentations de surfaces. Si cette représentation est formellement la plus précise il est bon de remarquer que, comme la géométrie constructive, elle élude en fait les calculs et qu'il faudra en général lui associer une représentation dérivée pour accélérer les performances des algorithmes. On ne pourra pas éviter de programmer les algorithmes d'intersection de surfaces. Notons cependant que beaucoup d'informations peuvent être calculées directement à partir de cette représentation, ainsi si nous connaissons un point de la courbe et si nous voulons la tangente à la courbe en ce point, il suffit de demander aux deux surfaces de produire les vecteurs normaux et d'en faire le produit vectoriel, le vecteur tangent étant dans l'intersection des deux plans tangents.

[Hoffman & Hopcroft] ont de plus fait remarquer que ce modèle pouvait conduire à des ambiguïtés dans la représentation d'un solide. Le cas qu'ils présentent est celui d'une surface qui s'auto-intersecte (nous avons remarqué que de telles surfaces pouvait être tolérées si les points singuliers ne sont pas à l'intérieur des faces). Si l'on définit une arête par l'intersection de cette surface et d'une autre surface on obtient plusieurs possibilités.

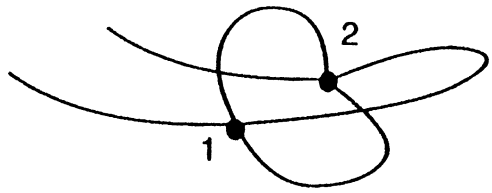
figure II-24 courbe ambiguë



surface avec auto-intersection



tore



intersection 3 courbes possibles entre les points 1 et 2

II-4-2-2 Equation paramétrique

Comme les surfaces les courbes peuvent être représentées par une équation paramétrique (x(t) , y(t) , z(t)). Les fonctions sont en général des polynômes où des fractions rationnelles (Béziers, B-Splines). Cependant la courbe d'intersection de deux surfaces ne rentre généralement pas dans ce cadre, il faut donc l'approximer. Deux options

se présentent en général quand on a calculé une série de points sur la courbe par un algorithme numérique.

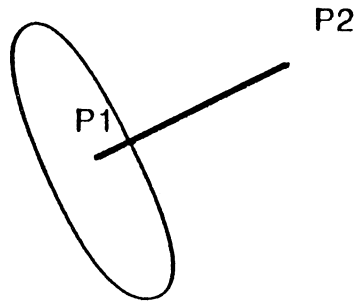
- Interpoler une courbe entre ces points, mais la courbe n'appartient pas en général aux surfaces.

- Représenter la courbe par une fonction (u(t) , v(t)) dans l'espace des paramètres de chacune des deux surfaces. Le problème est qu'il y a alors deux courbes, une sur chaque surface, qui ne coïncident que sur les points de départ de l'interpolation.

II-4-2-3 Représentation PVS

Pour les courbes simples (droites, cercles, coniques) on peut utiliser une représentation PVS à la place d'une équation (surtout pour le cercle qui nécessite des fractions rationnelles pour être exactement paramétré). Par exemple un cercle est représenté par deux points : le premier est le centre, le second indique l'axe du cercle et sa distance au premier point est le rayon.

figure II-25 représentation d'un cercle



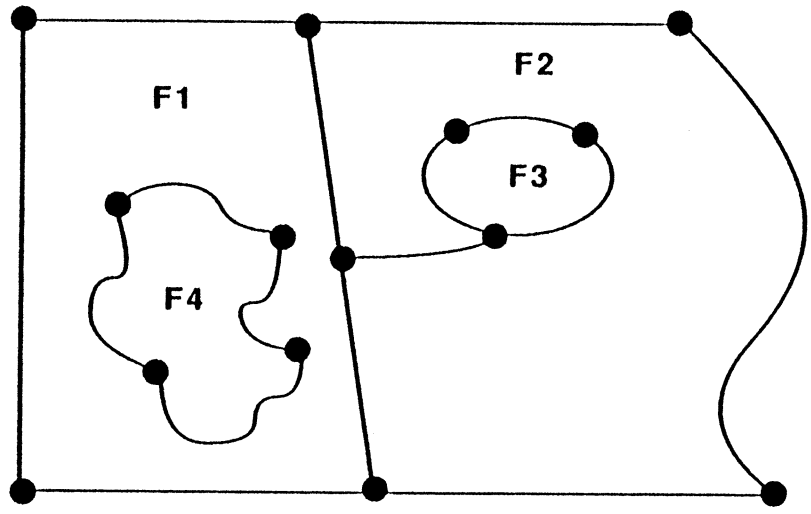
II-4-3 Représentation des faces

Nous allons maintenant montrer comment décrire une face qui est une partie d'une surface. Considérons un ensemble d'arêtes E et de sommets

V à chaque arête est associée une courbe support, à chaque sommet un point. Supposons de plus que les courbes et les points géométriques supports des éléments de V et E appartiennent tous à une même surface S. S est munie de son orientation naturelle et d'une topologie trace, c'est à dire que les voisinages d'un point de S sont les intersections avec S des sphères ouvertes centrées sur le point (voisinages dans l'espace Euclidien). Considérons l'ensemble F des points de la surface S appartenants aux arêtes tracées sur la surface, cet ensemble est évidemment fermé. Considérons l'ensemble complémentaire S - F qui est ouvert. Cet ensemble comporte plusieurs composantes connexes, dont une est non bornée, appelons régions ces composantes connexes. Nous pouvons donner la définition suivante :

Def. Une face est la fermeture d'une région bornée.

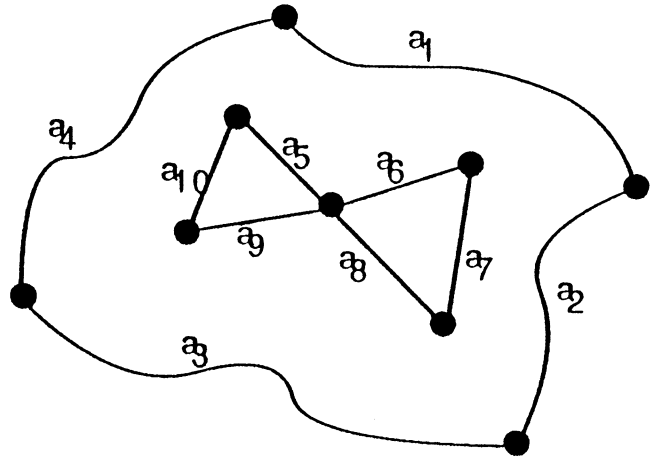
figure II-26 décomposition en faces



Notre problème est maintenant de trouver une représentation des faces en terme de structure de données. Pour cela nous allons utiliser le même principe que pour les solides, c'est à dire que nous allons décrire une face par sa frontière. On voit facilement que la frontière d'une face est formée d'un certain nombre de cycles, un cycle étant une chaîne

circulaire d'arêtes, au moins un cycle forme l' "extérieur" de la face, les autres sont d'éventuels "trous". Pour pouvoir déterminer en chaque point de la frontière où se trouvent l'intérieur et l'extérieur de la face nous allons convenir d'orienter les arêtes, les cycles deviennent alors des circuits.

figure II-27 une face de deux circuits : premier circuit a1 ... a4 deuxième circuit a5 ... a10

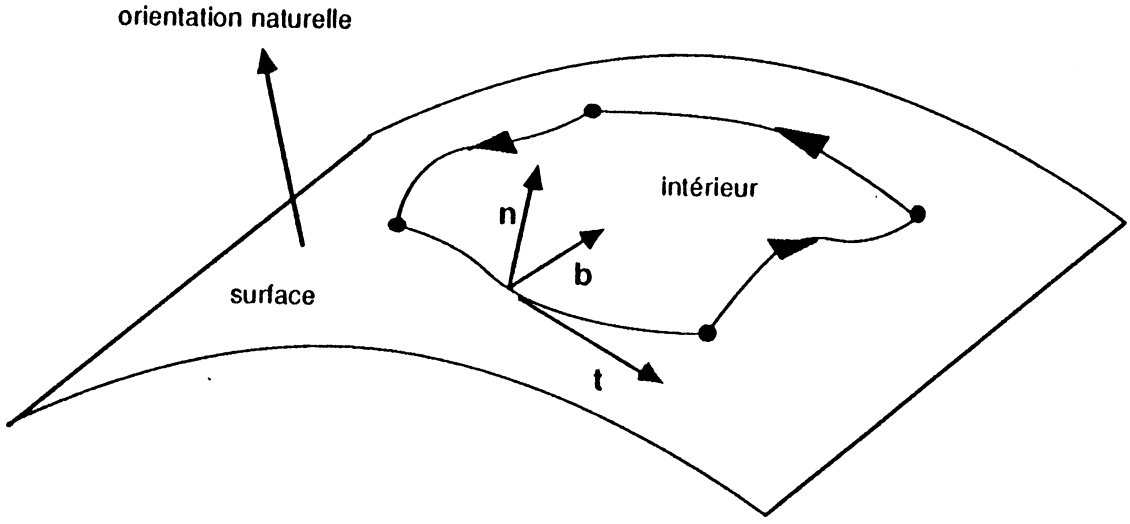


L'arête étant orientée il devient possible de définir en chaque point de la courbe un vecteur tangent orienté. On peut alors déterminer la position de la face par rapport à sa frontière en utilisant la règle des binormales :

Soient en un point de la frontière t le vecteur tangent et n le vecteur normal à la surface, nous définissons **b = n × t** leur produit vectoriel, b est le vecteur binormal à la courbe. Le vecteur binormal pointe vers l'intérieur de la face. Cette règle est compatible avec les règles d'orientation classiques : règle du bonhomme d'Ampère : un observateur se déplaçant sur la courbe dans le sens de son orientation avec la tête du coté de la normale à la surface, voit l'intérieur de la surface à sa gauche. Règle du tire-bouchon : si l'on imagine un tire-bouchon au milieu de la face et qu'on le fait tourner dans le sens d'orientation de la frontière (circuit exterieur) il se déplacera dans le sens de la normale, par contre les circuits des trous tournent en sens opposés.

Il se peut que l'orientation naturelle de notre surface ne corresponde pas à l'orientation désirée de la face, il nous faut donc préciser en associant la face à la surface si nous voulons conserver l'orientation naturelle ou adopter l'orientation opposée.

figure II-28 règle des binormales



En résumé la description d'une face se compose de la manière suivante:

- pointeur sur la description d'une surface (qui peut être commune à plusieurs faces)

- signe de la surface : orientation naturelle ou opposée.

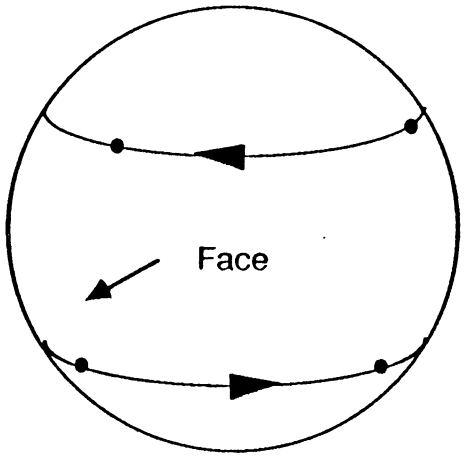
- ensemble fini de circuits décrivant la frontière :

Chaque circuit est une liste circulaire d'arêtes.

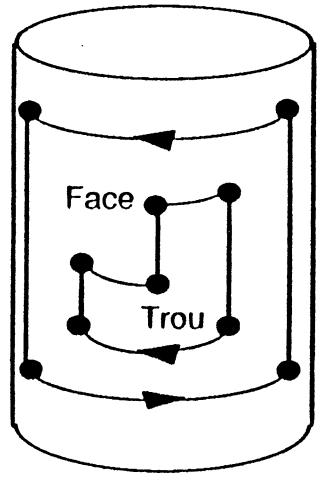
Les arêtes intervenant dans la description de la face doivent vérifier la condition suivante :

Les arêtes ne s'intersectent pas ailleurs qu'aux sommets extrémités.

figure II-29 exemples de faces



deux circuits sur une sphère
orientation naturelle
(vers l'exterieur de la sphère)



deux circuits sur un cylindre

Classification point - face

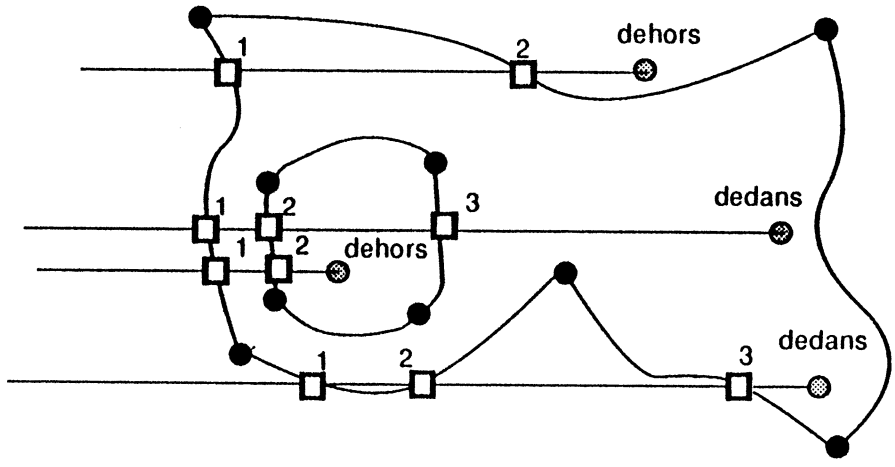
Nous allons justifier le fait que cette description est suffisante pour définir une face. Pour cela nous allons donner un algorithme qui sera capable à la donnée d'une face et d'un point de nous dire si le point est dans la face, hors de la face ou sur la frontière de la face. Ce problème est connu sous le nom de classification (in, on, out).

Si nous savons répondre à cette question notre description de la face est suffisante pour reconnaître l'intérieur et l'extérieur.

Une technique courante en deux dimensions consiste à tracer une demi-droite à partir du point et à compter le nombre d'intersection de cette demi-droite avec les arêtes de la face. Si ce nombre est pair (y compris

nul) le point est hors de la face, sinon il est dedans, il est sur la frontière si il appartient à une des arêtes.

figure II-30 algorithme de la demi-droite



Cet algorithme se prête à quelques critiques que voici :

a - Cas particuliers

La demi-droite peut rencontrer un sommet ou se confondre avec une arête. Pour cela, il faut complexifier l'algorithme soit en cherchant une autre demi-droite (au hasard) jusqu'à élimination des cas particuliers, soit en étudiant ces cas plus précisément. Cependant cet algorithme fonctionne bien pour les polygones du plan. Les critiques importantes sont les suivantes.

b - Orientation

L'algorithme ne tient aucun compte de la règle des binormales et donc de l'orientation des circuits de la face. En effet il considère la région finie délimitée par les arêtes. Le problème est encore plus net avec la remarque suivante.

c - Surfaces finies

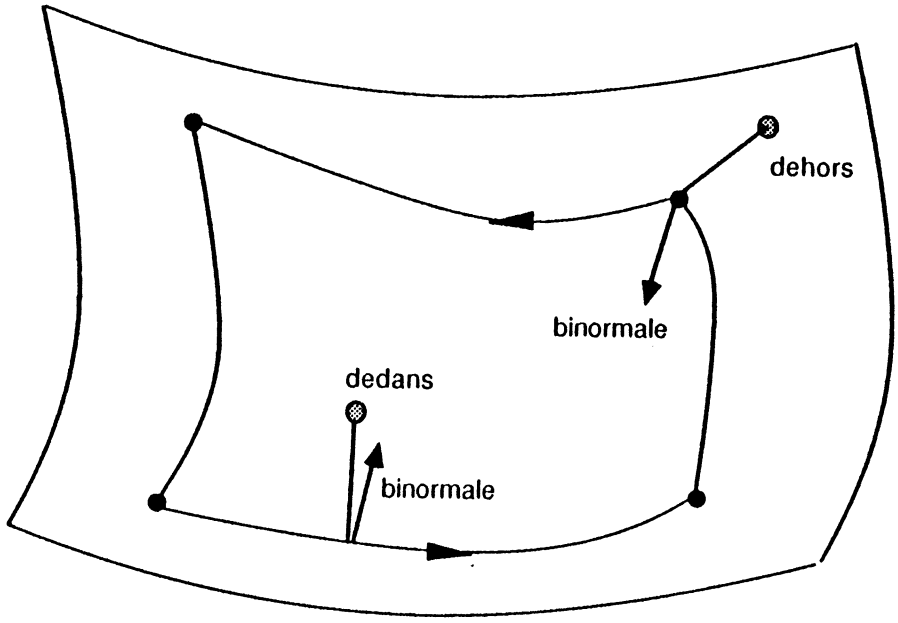
Cet algorithme n'est correct que si la demi-droite (on peut la remplacer par une demi-courbe sur une surface non plane) s'éloigne vers l'infini sur la surface. En effet la justification de l'algorithme est qu'au delà de toute intersection se trouvent des points nécessairement à l'extérieur de la face et que chaque intersection inverse l'état intérieur-extérieur des points de la courbe, ce qui permet de revenir jusqu'à l'état du point de départ.

Sur le plan nous savons que toute demi-droite nous mènera vers l'infini, mais ceci n'est pas toujours vrai. Sur le cylindre et le cône il nous faut choisir des demi-droites sur les méridiens de la surface, ce qui réduit les possibilités. Enfin sur les surfaces comme la sphère ou le tore qui sont bornées il n'y a pas d'infini. Dans ce cas un circuit partage la surface en deux régions bornées qui sont toutes deux des faces, et seule l'orientation du circuit permet de choisir entre elles.

Nous proposons donc un autre algorithme qui utilise l'orientation des arêtes. Le principe est de chercher l'élément de la frontière de la face (sommet ou arête) qui est le plus proche du point à classifier. En toute rigueur il faut utiliser des distances géodésiques sur la surface, mais celles-ci sont difficiles à calculer, nous les remplaçons donc par les distances euclidiennes en trois dimensions en sachant que pour la plupart des surfaces que nous utilisons les distances géodésiques et les distances euclidiennes croissent dans le même sens et donc que la plus courte nous donnera le même élément. Nous pouvons maintenant nous ramener à un problème restreint en considérant la région comprise dans une sphère centrée sur le point de rayon légèrement supérieur à la distance. Cette sphère ne contient que le point et l'élément. Il suffit maintenant de calculer une binormale sur l'élément (si c'est un sommet on calcule la somme des binormales aux arêtes incidentes) puis de faire le produit scalaire de cette binormale avec un vecteur allant de l'élément

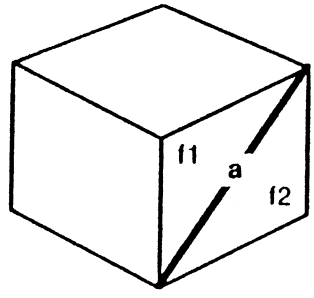
vers le point. Si le produit scalaire est positif le point est dedans, négatif le point est dehors, nul le point est dessus.

figure II-31 classification avec les binormales



Remarquons que la notion de face que nous venons de décrire peut différer de celle que nous avions définie précédemment, ceci à cause des arêtes fantômes, ainsi sur la figure suivante l'arête a sépare deux faces là où l'intuition n'en mettrait qu'une :

figure II-32 face séparée par une arête fantôme



Cette remarque montre qu'en plus des permutations de la structure de données qui sont facilement détectables il peut etre difficile de décider de l'égalité de deux solides et donc que l'unicité du modèle frontière n'est pas assurée.

II-4-4 Conditions de validité d'un modèle frontière

Un modèle frontière d'un solide est un ensemble de faces, éventuellement regroupées en nappes. Nous supposerons que chacune de ces faces est valide, c'est à dire qu'elle respecte la représentation du chapitre précédent et que particulièrement les arêtes ne s'intersectent pas hors des sommets.

Alors pour que l'ensemble de ces faces forment la frontière d'un solide il faut respecter trois conditions qui expriment que l'ensemble des faces est une surface fermée orientable et 2-manifold.

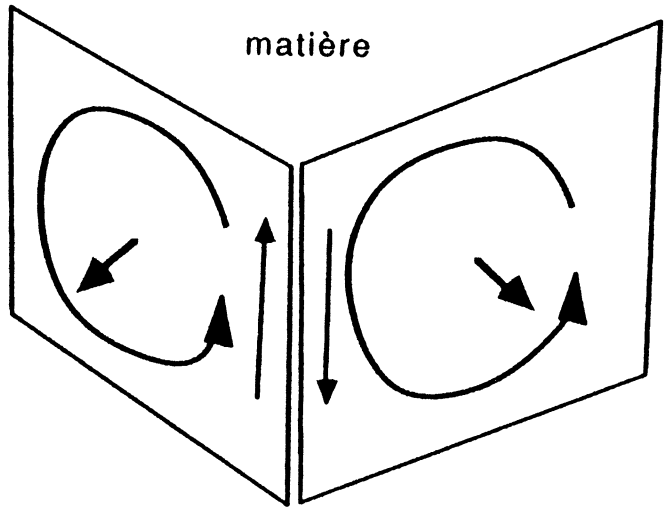
- Les faces ne s'intersectent pas hors des arêtes.

Ceci nous garantit que la surface du solide ne présente pas d'auto-intersections.

- Chaque arête appartient à deux faces qui la traversent dans des sens opposés.

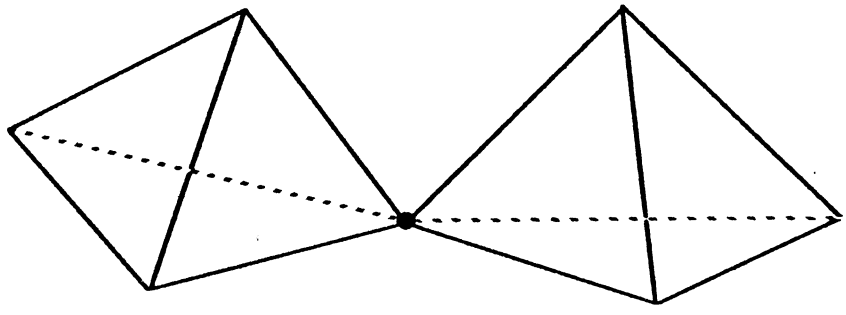
Cette condition connue sous le nom de **règle de Moëbius** assure que la surface est fermée et orientable. On en voit l'illustration graphique sur la figure suivante qui montre bien que les orientations des deux faces sont compatibles.

figure II-33 règle de Moëbius



- Sur chaque sommet les faces doivent former un ensemble connexe pour la relation de partage d'arête. Cette condition assure que la surface est 2-Manifold au sommet, on voit sur la figure suivante le cas typique de non respect de cette condition.

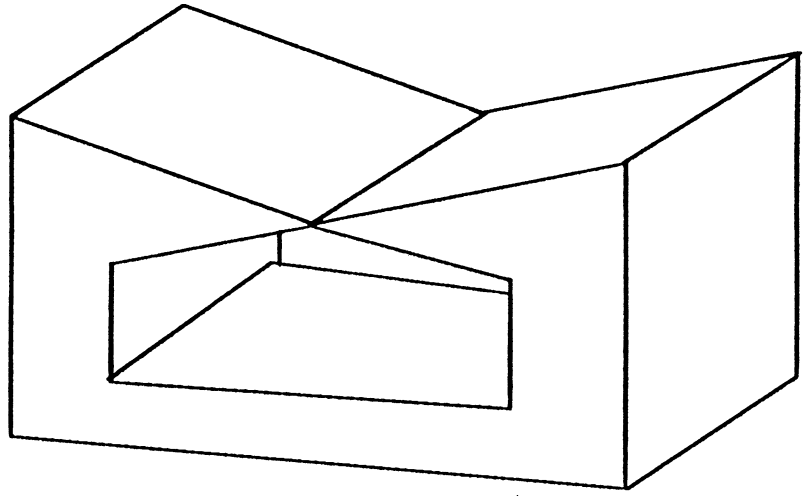
figure II-34 polyèdre non 2-manifold



En fait la controverse subsiste au sujet de cette condition et d'une forme affaiblie de la précédente qui permettent de représenter des objets non 2-manifolds **[Hoffman & Hopcropft]**. La dernière condition peut être omise, et dans la deuxième on peut proposer que chaque arête appartiennent à un nombre pair de faces correctement orientées, ce qui

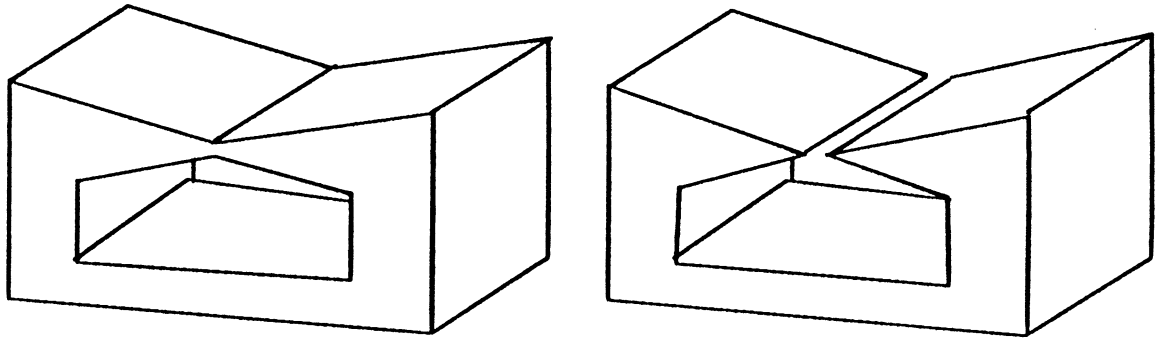
permet de prendre en compte des solides comme celui de la figure suivante :

figure II-35 arête à plus de 2 faces



On peut en fait traiter de tels solides avec les conditions 2-manifold, il suffit de dédoubler l'arête partagée par quatre faces et de choisir entre les deux interprétations suivantes :

figure II-36 deux manières de rendre 2-manifold



Remarquons que l'interprétation de droite est plus satisfaisante si l'on imagine un solide matériel et que l'on envisage sa fragilité.

Le coût informatique est faible pour vérifier les deux dernières conditions car ce sont des propriétés topologiques. Par contre pour vérifier la première condition il faut calculer des intersections de surfaces, si on peut l'envisager pour un solide polyédrique (mais c'est déjà très couteux) cela devient prohibitif pour un polyèdre topologique. Ces calculs ne sont en général pas mis en oeuvre dans les systèmes de C.A.O ce qui signifie que la validité des objets n'est pas garantie, à la charge des procédures de création de solides de produire des résultats corrects. Cependant il arrive dans tous les grands systèmes de C.A.O que des objets corrompus subsistent dans les bases de données Il pourrait être intéressant de disposer d'algorithmes de vérification, réparation qui pourraient tourner en différé sur les bases de données.

II-4-5 Formules d'Euler

Les propriétés topologiques des graphes associés aux solides permettent de dériver certaines formules reliant les nombres de faces, arêtes, sommets. Ces formules sont connues sous le nom de formules d'Euler-Poincaré, on peut consulter **[Wilson 85]**. Ces formules ne sont pas vérifiées pour un solide incorrect, mais par contre elles ne garantissent pas du tout la validité d'un solide si elles sont vérifiées.

La première formule est la suivante :

$$v - e + f = 2$$

Où v est le nombre de sommets, e le nombre d'arêtes, f le nombre de faces, par exemple pour un cube : 8 - 12 + 6 = 2. Cette formule n'est valable que pour un solide de genre 0, connexe, dont les faces ne sont pas trouées. Nous ne donnons pas la démonstration qui peut etre trouvée dans l'ouvrage de Lakatos **[Lakatos]**.

Si il y a plusieurs solides on peut écrire :

$$v - e + f = 2s$$

Où s est le nombre de solides.

Si l'on veut prendre en compte le genre du solide, c'est à dire le nombre de trous qui le traversent on obtient une formule un peu modifiée :

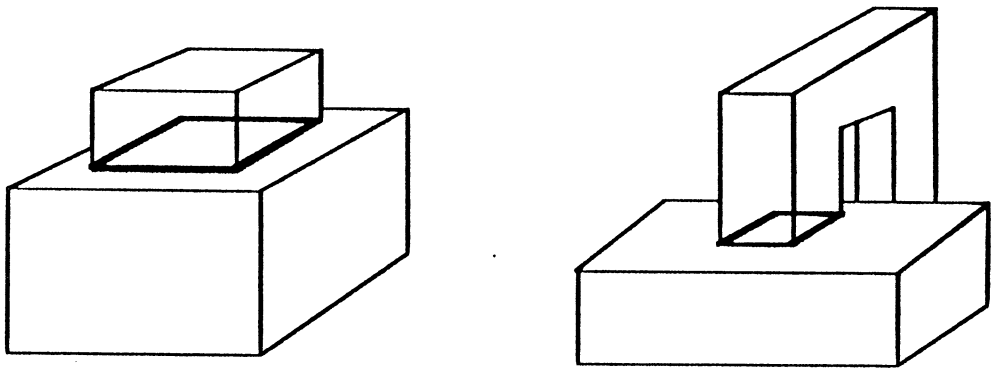
$$v - e + f = 2(1 - g)$$

Où g est le genre du solide. Pour un solide formé de plusieurs nappes (solide avec cavités, plusieurs solides) on écrit :

$$v - e + f = 2(s - g)$$

Où s est le nombre de nappes et g la somme des genres des nappes. Il nous reste à prendre en compte les faces trouées pour obtenir une formule complète. Nous pouvons voir sur la figure suivante que l'on peut supprimer un trou dans une face en ajoutant une nouvelle face et que ceci à pour effet d'augmenter le nombre de nappes (cas 1 figure de gauche) ou de diminuer le genre en supprimant un trou (cas 2 figure de droite) soit dans tous les cas d'ajouter 2 au terme droit de la formule.

figure II-37 effet du remplacement d'un trou par une face



On peut donc proposer une formule définitive :

$$v - e + f - h = 2(s - g)$$

Où h est le nombre de trous dans des faces, on remplace parfois **f - h** par **2 f - l** ou **l** est le nombre total de circuits, ce qui revient au même puisque **l = h + f.**

II-4-6 Structures de données

Nous allons maintenant discuter les structures de données utilisées pour la représentation des solides. Nous allons nous limiter à la représentation frontière. Plus précisément nous allons donner des structures de données pour représenter les informations topologiques associées à un solide, c'est-à-dire faces, arêtes et sommets. Les informations géométriques posent moins de problème puisqu'il s'agit en général d'informations scalaires : coordonnées, coefficients d'équations.

Nous allons commencer par recenser les opérations d'accès à la structure de données, aussi bien pour l'interrogation que pour la création. Pour la création de structure de donnée une approche intéressante est celle des opérateurs Eulériens qui permettent de définir une interface indépendante de la structure de données avec l'application qui construit les solides **[Eastman & Weiler], [Mantyla 82, 84].**

II-4-6-1 Opérateurs Eulériens

Les opérateurs Eulériens sont des primitives de manipulation de structure de données (création, destruction, modification) qui portent ce nom car ils sont construits à partir de la formule d'Euler. En effet chaque opérateur Eulérien crée ou détruit certains éléments de la structure de données (face, arête, sommet, trou, nappe) en assurant que la formule d'Euler est toujours valide. Par exemple examinons cette formule :

$$v - e + f - h = 2 (s - g)$$

Et imaginons que nous voulions créer une face. Nous ne pouvons créer une face seule, par contre nous pouvons créer une face et une arête en

même temps puisque **f - e** sera nul, nous pouvons aussi créer une face et un trou **f - h** sera nul. Les opérateurs Eulériens portent des noms mnémotechniques formés à partir des lettres suivantes :

m	make	créer
k	kill	détruire
s	shell	nappe
v	vertex	sommet
e	edge	arête
f	face	face
h	hole	trou
g	genus	genre

Ainsi l'opérateur qui crée une arête et une face sera nommé mef pour make edge face. Nous donnons dans la table suivante une liste d'opérateurs Eulériens de base empruntée à Mantyla, avec leur action sur les composants de la formule d'Euler :

V	E	F	H	S	Opérateur	
1	0	1	0	1	mvfs	make vertex, face, shell
1	1	0	0	0	mev	make edge, vertex
0	1	1	0	0	mef	make edge, face
-1	0	-1	0	-1	kvfs	kill vertex, face, shell
-1	-1	0	0	0	kev	kill edge, vertex
0	-1	-1	0	0	kef	kill edge, face

Remarques

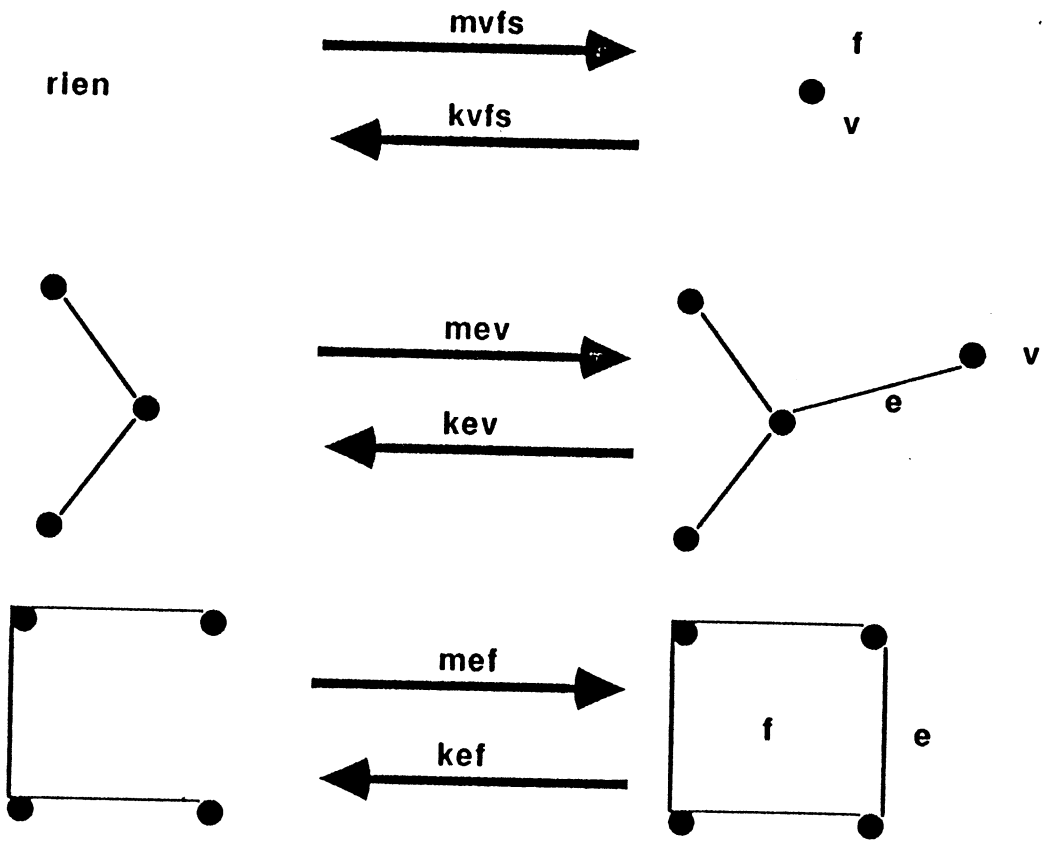
- Les opérateurs **kvfs, kev, kef** sont les inverses exacts de **mvfs, mev, mef.**

- L'opérateur **mvfs** est utilisé pour initialiser toute construction de nappe puisqu'il est le seul à en créer, cette construction démarre par la nappe la plus simple qui contient une face réduite à un sommet.

Nous n'avons pas donné d'opérateurs pour modifier le genre, on peut consulter **[Eastman & Weiler]**. De même pour les trous on peut introduire des opérateurs qui transforment les faces en trous en modifiant le nombre de nappes ou le genre.

L'emploi correct de ces opérateurs nécessite bien sur de donner des arguments qui désignent les objets que l'on veut créer ou détruire et qui permettent de lever certaines ambiguïtés. Les opérateurs Eulériens de la table sont illustrés sur la figure suivante.

figure II-38 Opérateurs Eulériens



Les opérateurs Eulériens ne sont pas encore largement utilisés dans les systèmes de modélisations solides, ils permettraient cependant de

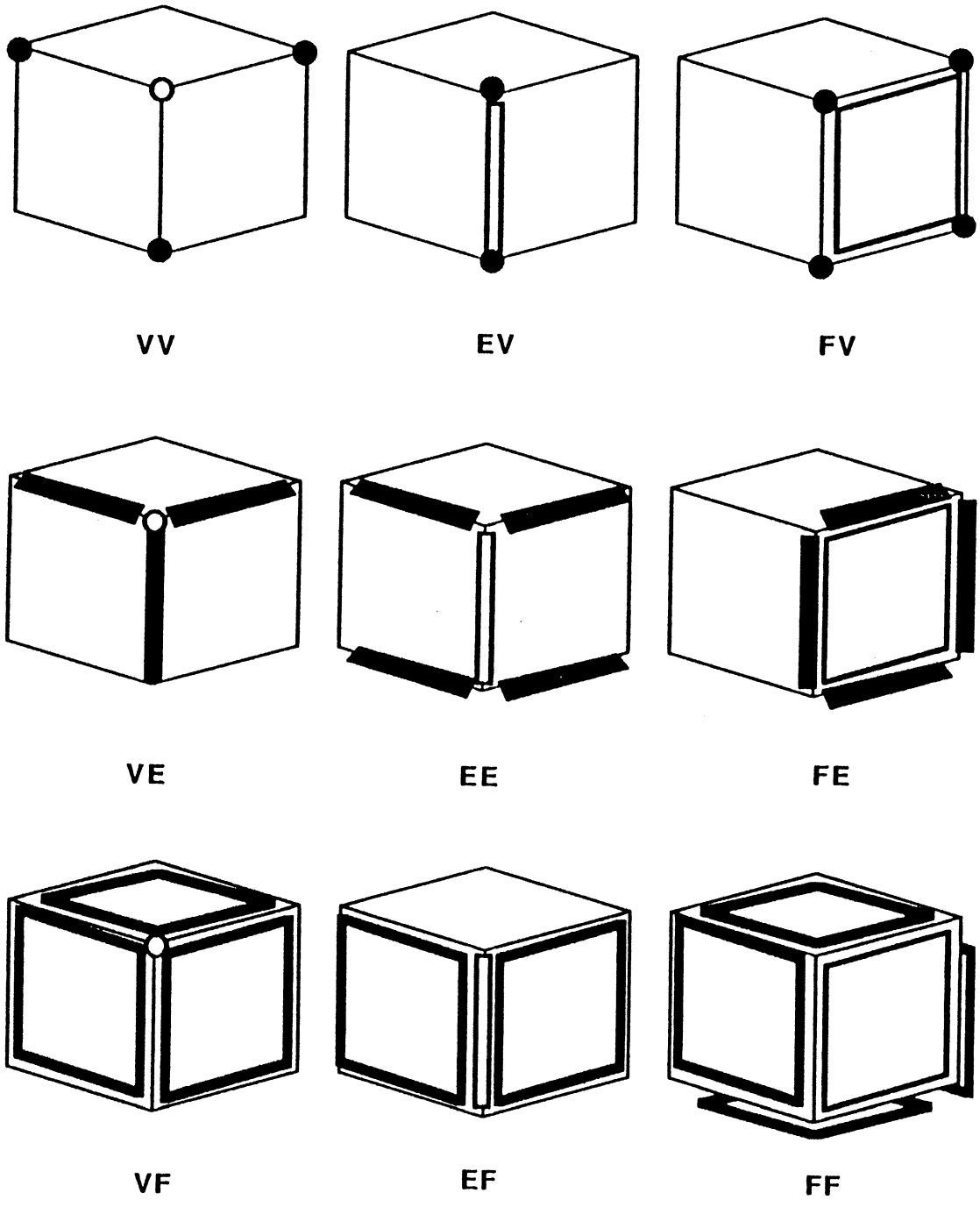
donner des algorithmes de construction de solides indépendants de la structure de donnée. Cependant ils n'existe actuellement aucun standard et les auteurs diffèrent sur le choix des opérateurs fondamentaux et sur leurs spécifications.

II-4-6-2 consultation de la structure de donnée

Les structures de données topologiques sont constituées d'un ensemble de relations entre objets et les interrogations portent sur ces relations. Ainsi des questions typiques seraient : quelles sont les faces contenant telle arête ? Quelles sont les faces voisines de telle face ? Quels sont les sommets extrémités de telle arête ? La figure suivante illustre ces différentes relations entre les entités fondamentales sommets, arêtes, faces. Nous conviendrons de noter avec deux lettres une relation : ainsi **VF** signifie relation sommet - face, soit partant d'un sommet retrouver les faces qui lui sont adjacentes. (cf **[Weiler 85]**)

Remarquons que certaines de ces relations lient toujours le même nombre d'éléments ainsi une arête possède deux sommets et appartient à deux faces, alors que d'autres sont variables : un sommet appartient à plusieurs faces, éventuellement ordonnées. L'étude des relations les plus utilisées par une application donnée permet de choisir la structure de donnée la mieux adaptée.

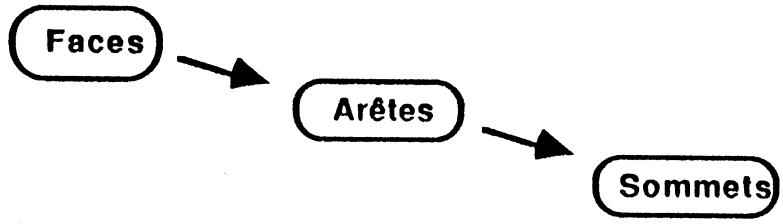
figure II-39 les neuf relations



II-4-6-3 Evaluation de la structure de données

Le choix d'une structure de données fait entrer en compte deux critères contradictoires : la rapidité d'accès et l'encombrement mémoire (cf **[Woo 85]**) chacune des neuf relations fondamentales proposées au chapitre précédent peut être mémorisée directement ou indirectement dans ce dernier cas il faut prévoir un algorithme pour retrouver la relation à partir des autres. Ainsi, si l'on a mémorisé la relation face-arêtes FE et la relation arête-sommets EV, la relation transitive face-sommets FV est facile à reconstituer et ne doit pas du tout être mémorisée. Par contre pour avoir la relation arête-faces EF il faut parcourir la relation FE ce qui coûte un temps propositionnel au nombre de faces, si cette relation est beaucoup utilisée il peut être avantageux de sacrifier de l'espace mémoire pour la mémoriser directement. Le coût d'une structure de données en encombrement mémoire et en temps d'accès pour chaque relation peut s'évaluer facilement en fonction des nombres de sommets, arêtes et faces du modèles soient v, e et f.

Par exemple imaginons la structure de données la plus simple qui consiste à mémoriser les relations FE et EV.



L'encombrement mémoire est le suivant :

- relation FE : Pour une face chaque pointeur vers une arête occupe un emplacement mémoire, si l'on somme sur l'ensemble des faces on obtient deux fois le nombre d'arêtes puisque chaque arête appartient à deux faces : soit 2 e.

- relation EV : A chaque arête sont associés deux sommets soit deux emplacements mémoires, encore 2 e.

L'encombrement mémoire est 4 e. Remarquons que nous ne comptons que la structure de données topologiques et pas les données géométriques qui sont indépendantes du choix de la structure de données.

Le coût d'accès de chaque relation est donné par la table suivante.

VV	e / 2	Se déduit de VE et EV.
VE	e / 2	Inverser EV.
VF	f / 2	Inverser FV.
EV	direct	
EE	f / 2	Par EF et FE.
EF	f / 2	Inverser FE.
FV	direct	Par FE et EV.
FE	direct	
FF	f / 2	Par FE et EF.

Cette table signifie que, par exemple, pour trouver toutes les faces voisines d'une face donnée (FF) il faut chercher les arêtes de la face (FE) ce qui est direct, puis explorer les faces pour trouver l'autre face qui contient l'arête, ce qui coûte f / 2. (en moyenne).

Remarquons que la structure de données doit évidemment contenir assez d'informations pour pouvoir reconstituer les neuf relations.

De plus certaines relations sont ordonnées (arêtes autour d'une face par exemple) et cet ordre doit être préservé pour que la description soit correcte. Nous verrons dans la partie III que la relation VV si l'ordre des sommets est respecté suffit aussi à décrire un solide.

Au sujet des mesures de coût en fonction des nombres de sommets de faces et d'arêtes on a souvent coutume de se ramener à un seul

paramètre qui est le nombre de faces du solide, c'est le plus intuitif. Le nombre de sommets et d'arêtes peuvent s'approximer simplement à partir du nombre de faces en admettant qu'il y a environ quatre arêtes par faces et entre trois et quatre faces par sommets, ceci est d'autant plus vrai pour les grands modèle qui sont des approximations polyédriques de surfaces par des panneaux rectangulaires. Si l'on admet quatre arêtes par face et par sommet on peut déduire qu'il y a deux fois plus d'arêtes que de faces ou de sommets, soit e = 2 f et s = f. La pratique montre qu'en fait le nombre d'arêtes par face est plus élevé et donc que le nombre d'arêtes croit un peu plus vite que deux fois le nombre de faces.

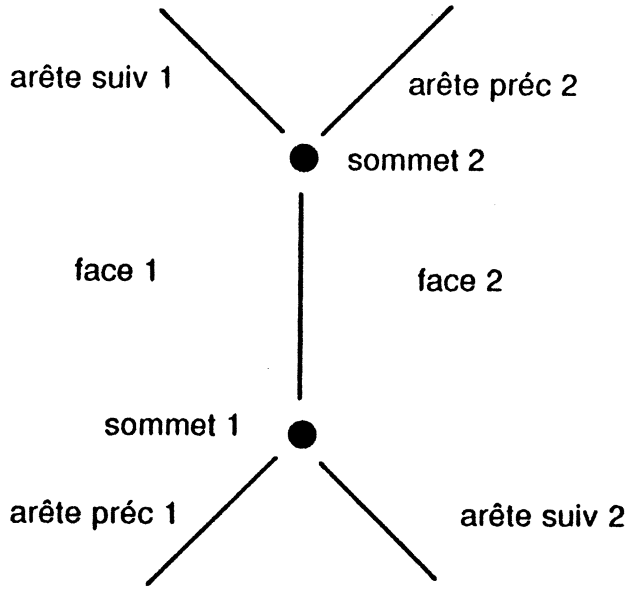
Nous allons maintenant décrire quelques structures de données : la "winged-edge structure" qui est très répandues dans la communauté universitaire et celle que nous avons utilisée qui dérive de celle du système de C.A.O EUCLID. On trouveras d'autre structures dans **[Ansaldi & all.], [Mantyla], [Wilson], [Woo].**

II-4-6-4 Structure de données Winged-edge

Cette structure de données est assez répandue dans la communauté universitaire, elle a été originellement développée par Baumgart **[Baumgart], [Weiler 85]**. C'est une structure de données uniforme qui repose sur un seul type d'enregistrement associé à chaque arête. Cet enregistrement contient huit champs qui sont : les deux sommets de l'arête, les deux faces incidentes à l'arête, les quatre arêtes qui suivent l'arête sur ces deux faces. La figure suivante illustre cette structure de donnée et justifie son nom (winged-edge = arête ailée).

Sommet 1 et sommet 2 sont les deux extrémités de l'arête, on convient que l'ordre sommet 1, sommet 2 correspond au sens de parcours de l'arête dans la face 1, sur la face 1 l'arête est précédée par arête préc 1 et suivie par arête suiv 1, il en est de même pour la face 2.

figure II-41 structure winged-edge



sommet 1	sommet 2
face 1	face 2
arête préc 1	arête préc 2
arête suiv 1	arête suiv 2

Cette structure de donnée est relativement encombrante (8 e) par contre toutes les relations peuvent être consultées en temps constant à partir du moment où nous connaissons une arête. Par exemple imaginons que nous voulons toutes les faces incidentes à un sommet et que nous connaissons une arête incidente à ce sommet, cette arête nous donne deux faces incidentes au sommet et il est facile en partant de l'une de parcourir les faces incidentes en s'arrêtant des que l'on rencontre la face

de départ de nouveau. Remarquons bien qu'il faut toujours connaître une arête pour pouvoir explorer la structure de donnée.

Nous n'avons pas utilisé cette structure de données car nous voulions représenter des solides mais aussi des vues bidimensionnelles et des fils-de-fer, or cette structure de données ne permet pas de mélanger différents types d'objets. De plus, ce travail étant mis en oeuvre en collaboration avec le système EUCLID il fallait que les structures de données soit compatibles.

II-4-6-5 Structure de données d'EUCLID

La structure de données d'EUCLID repose sur trois tables [Euclid] : la table des points qui contient les coordonnées des sommets (comme dans toutes les structures de données géométriques) un point est repéré par son indice ou pointeur dans cette table, la table des liaisons qui contient des pointeurs vers la table des sommets, et la table des éléments qui contient des pointeurs vers les trois tables. Un élément de cette dernière table appelé géométrique , est constitué de trois champs :

- Un champ de type qui indique à quel objet géométrique on a affaire, ce peut être un sommet, une ligne brisée, une face, un solide. Ou encore d'autres objets : courbes, surfaces etc ...

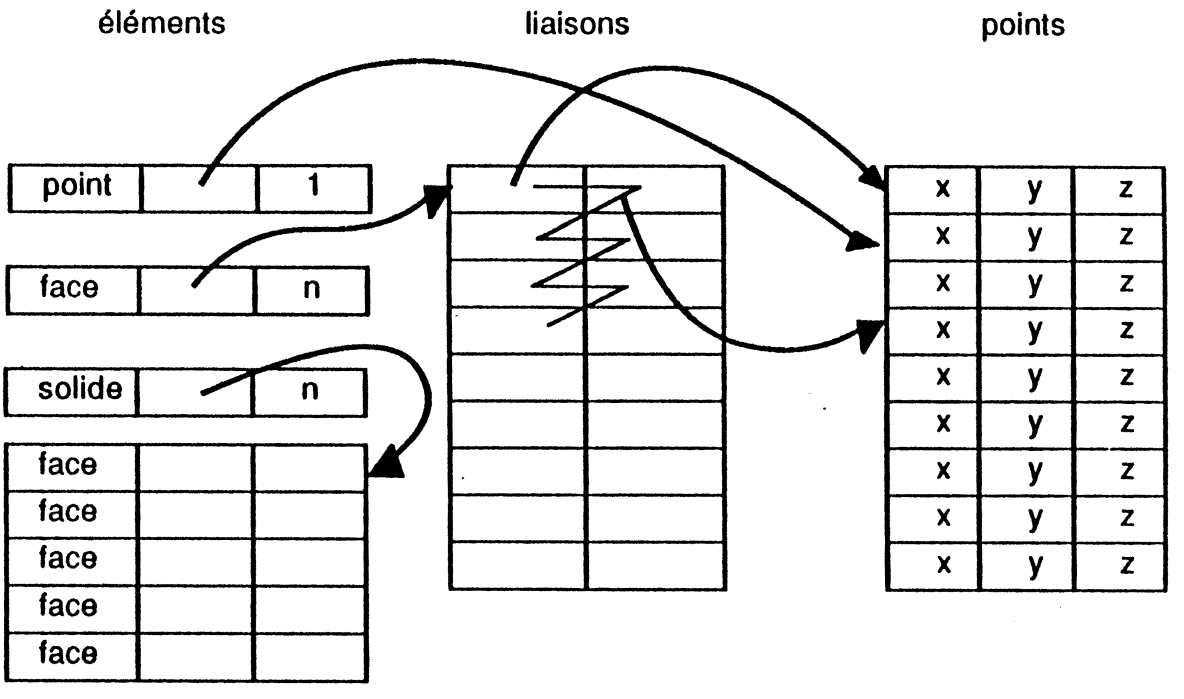
- Un champ pointeur qui est un indice dans une des trois tables et qui indique le premier des éléments désignés par le géométrique.

- Un champ nombre qui indique combien d'éléments il faut prendre en compte à partir du premier.

Ainsi, un point contient un pointeur vers la table des points et un nombre toujours de un, une face contient un pointeur vers la table des liaisons et l'on trouvera à cet endroit les indices des points qui forment la face (face polygonale). Un solide contient un pointeur vers un groupe de faces qui

se trouvent dans la table des éléments. Tout ceci est illustré sur la figure suivante.

figure II-41 structure de données d'EUCLID



Il s'agit d'une structure de données efficace et compacte, même avec la contrainte de regroupement des éléments, elle est de plus souple grâce au mécanisme de typage qui permet de mélanger différents objets dans la même structure. Cependant nous remarquerons qu'il s'agit d'une structure hiérarchique descendante et que donc certaines relations sont couteuses à obtenir. De fait la relation Face-Face qui permet de trouver les faces adjacentes à une face donnée sur un solide est mémorisée directement pour accélerer les algorithmes.

II-4-6-6 Structure de données utilisée

La structure de donnée que nous utilisons dans la mise en oeuvre des algorithmes qui sont présentés dans ce travail est construite autour de celle d'EUCLID puisque les algorithmes doivent s'intégrer étroitement dans ce système qui fournit les entrées et récupère les résultats. Il s'agit en fait d'une extension temporaire de cette structure de données qui répond aux trois bèsoins suivants :

- Pouvoir travailler sur un nombre d'éléments réduits, nous dupliquons une partie de la structure de données du système de CAO pour pouvoir manipuler des tables qui ne contiennent que les éléments d'intérêt et non pas tous les objets crée dans le système depuis le début de la session.

- Inverser certaines relations, nous avons dans ces algorithmes un grand besoin d'informations inverse de l'ordre hiérarchique, c'est à dire : étant donné un sommet, quelles arêtes le contiennent, étant donné une arête, quelles faces la contiennent ... Pour cela nous avons mis en place une structure de données pour mémoriser ces relations inverses et ne pas avoir à les recalculer.

- Représenter des solides à arêtes courbes, la structure de données décrite ci-dessus ne permet de représenter que des polyèdres, c'est-à-dire des solides dont les faces sont planes et donc les arêtes rectilignes. Hors nous avons utilisé des solides dont les faces se trouvent sur des surfaces courbes (essentiellement des quadriques de révolution), il faut donc que notre structure de données puisse intégrer de telles surfaces et les arêtes courbes.

Notre structure de données temporaire (à la fin des calculs les solides obtenus sont remplacés par une approximation polyédrique) est donc constituée de la manière suivante : elle n'est pas typée mais pour chaque type d'objet il existe une table dédiée, si l'on perd en souplesse on gagne cependant en rapidité. Les différentes tables sont :

- Listes : Les listes ne sont pas les objets géométriques mais une structure chaînée pour associer des listes à des objets, par exemple quand nous voulons inverser une relation temporairement. Cette structure est formée de deux tables fils et frère, le champ fils contient l'indice du premier élément de la liste, le champ frère contient l'indice de la liste des autres éléments, la liste vide est un zéro. Ces listes sont utilisées de façon très dynamique et nous avons ajouté un mécanisme de récupération de la place des listes périmées (le FORTAN ne permet pas la gestion dynamique de l'espace). Nous n'avons pas utilisé les listes toutes les fois que nous voulions associer plusieurs objets à un seul (voir les circuits plus bas), mais quand les ensembles sont petits ou construits de façon dynamique avec des modifications fréquentes.

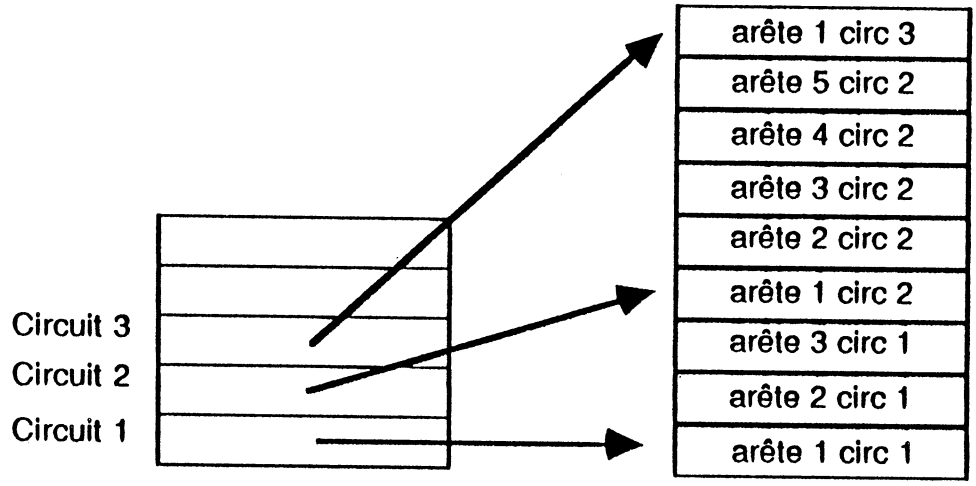
- Sommets : c'est une table d'éléments géométriques du type point EUCLID, ce sont donc des pointeurs vers la table des points. Chaque sommet à un indice dans cette table et nous faisons attention qu'aucun point ne se trouve deux fois dans la table (indices différents mais coordonnées identiques).

- Arêtes : c'est la table des arêtes elle contient d'une part les indices des deux sommets extrémités de l'arête, toutes les arêtes sont implicitement orientées, et d'autre part la réalisation géométrique de l'arête qui est un élément de la structure de données EUCLID. Remarquons que le système permet d'utiliser des courbes et en particulier des arcs de cercles.

- Surfaces : Cette table recense toutes les surfaces du système, comme EUCLID n'utilise pas la représentation PVS que nous avons choisie cette table contient pour chaque surface : son type parmi plan, cylindre, cône, sphère, tore et sa représentation PVS sous forme d'un élément géométrique EUCLID et d'un réel comme nous l'avons déjà montré en II-4-1-3.

- Circuits : Les circuits sont les éléments qui constituent les frontières des faces, un circuit et formé par un ensemble d'arêtes. Pour représenter ces

ensembles nous utilisons deux tables, l'une associe à chaque indice de circuit un pointeur dans la deuxième table qui contient les indices des arêtes, comme on le voit sur la figure suivante. Ces indices d'arêtes sont signés pour tenir compte de l'orientation de l'arête dans le circuit, si l'indice est positif l'arête conserve son orientation, si il est négatif l'arête fait partie du circuit dans le sens opposé à son orientation.



- Faces : Cette table associe à chaque face un indice entier en la décrivant par :

L'indice de la surface qui contient la face, la description des circuits qui forment la face soit l'indice d'une liste de circuits, le premier circuit est l'extérieur, les autres sont des trous (sur une surface fermée comme la sphère il n'y aurait même plus lieu de distinguer l'extérieur des trous).

Nous verrons dans la description des algorithmes (partie IV) que nous construisons de nombreuses autres tables temporaires pour inverser certaines relations, par exemple pour chaque sommet la liste des arêtes incidentes etc...

Remarquons qu'il n'y a pas de table des solides, en effet notre algorithme ne construit a priori qu'un seul solide qui est formé par les faces.

III - Une approche topologique

Nous allons envisager dans cette partie une approche purement topologique du problème (cf **[Hanrahan]**), c'est à dire que nous n'utiliserons aucune information géométrique. Nous avons choisi de ne nous intéresser ici qu'au problème du fil-de-fer, c'est-à-dire : étant donné un fil-de-fer tridimensionnel, trouver tous les solides admettant ce fil-de-fer. La structure topologique associée au fil-de-fer est le graphe, pour le solide la structure est plus riche : c'est celle du graphe immergée dans une surface, le problème est de reconstituer la structure de faces, c'est à dire de trouver un ensemble de circuits qui vérifient les deux conditions que nous avons déjà énoncées.

- Chaque arête appartient à deux faces dans des sens opposés. (1)

- Les faces forment un ensemble connexe autour de chaque sommet. (2)

Nous allons d'abord donner une solution théorique du problème due à Edmonds **[Edmonds]** qui permet d'énumérer toutes les solutions du problème.

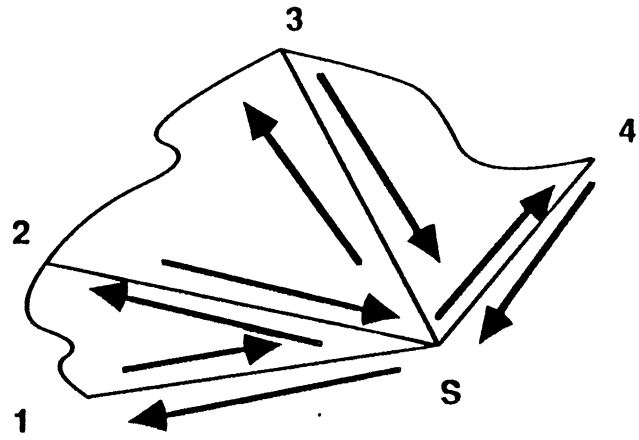
Puis nous verrons le cas particulier des graphes planaires (c'est à dire immergeables dans le plan) pour lequel des algorithmes très performants peuvent etre donnés.

III-1 Technique de permutation d'Edmonds

Cette technique repose sur la remarque suivante :

- Pour un solide les sommets voisins d'un sommet donné peuvent etre ordonné en permutation circulaire. Soit **S** un sommet du solide et **S'** un de ses voisins. L'arc **S'S** appartient à une seule face (propriété (1)) soit **S"** le sommet qui suit **S** sur le bord de cette face. On pose **f(S') = S"** , **f** est une permutation des voisins de **S**, le fait que **S** soit un cycle est une conséquence de la propriété (2), en effet chaque orbite de **f** correspond à une composante connexe des faces autour de **S**. Ces définitions sont illustrées sur la figure suivante.

figure III-1 permutation des voisins d'un sommet



f(1) = 2
f(2) = 3
f(3) = 4
f(4) = 1

Supposons maintenant que nous ne connaissons que le graphe **G** et la permutation des voisins associée à chaque sommet. Edmonds a montré que cette information était suffisante pour reconstituer l'ensemble des faces du solide et que ces faces vérifiaient les conditions (1) et (2). Ce résultat est assez évident, on reconstitue les faces de la manière suivante :

Nous nous donnons un graphe **G = (S, A)**, **S** est l'ensemble des sommets, **A** celui des arêtes. A chaque sommet V_i est associé une permutation circulaire f_i agissant sur l'ensemble des voisins de V_i.

Nous pouvons définir une permutation **p** sur l'ensemble des arcs de **G**, de la manière suivante :

$$f(V_i\ V_j) = V_j\ f_j(V_i)$$

Les orbites de cette permutation **p** forment des circuits avec les arcs du graphe. Chacun de ces circuits est une face et les propriétés (1) et (2) sont automatiquement vérifiées par ces faces. Remarquons qu'il n'y a qu'un circuit par face. Ce qui signifie qu'une face trouée (formée d'au moins deux circuits) sera considérée comme un ensemble de faces (coplanaires si il s'agit d'un solide polyédrique).

Nous avons ainsi montré que chaque solide associé au graphe était isomorphe à un ensemble de permutations sur les voisins de chaque

sommet. Il y a donc exactement autant de solides associés au graphe que de tels ensembles de permutations, ce qui peut se compter. Soit d_i le degré (nombre de voisins) du sommet S_i , le nombre de permutations circulaires des sommets voisins de S_i est $(d_i - 1)!$. Donc le nombre de solides admettant le graphe G comme fil-de-fer est :

$$(d_1-1)\,!\ .\ (d_2-1)\,!\ .\ \ldots\ (d_n-1)\,!$$

Il s'agit d'un nombre très grand, prenons par exemple le graphe d'un cube formé de 6 sommets de degré 3, on aurait $(2!)^6 = 64$ solutions.

Rappelons cependant que géométriquement la plupart de ces solides ne serons pas corrects à cause des intersections de faces. Cette remarque montre les limites de l'approche topologique dans le cas général.

Il est possible de restreindre le nombre de solides solutions en considérant le genre. En effet nous pouvons calculer le genre de chacun des solides par la formule d'Euler (il s'agit en fait du genre de la surface frontière du solide). L'énumération des permutations nous permettrais de calculer le genre minimum et maximum des solides associés au graphes, tous les genres intermédiaires sont d'ailleurs réalisés **[White]**. On sait que l'on appelle genre du graphe le genre minimum qui est le plus petit genre d'une surface sur laquelle il est possible de dessiner le graphe sans intersections d'arêtes. Nous pourrions ne considérer que les solides de genre minimum, et c'est ce que nous allons faire par la suite.

III-2 Graphes planaires

Nous allons maintenant considérer des graphes planaires, c'est-à-dire des graphes de genre 0, qui peuvent être dessinés sur le plan ou la sphère sans intersection d'arêtes. Nous allons voir que pour ces graphes la solution du problème peut être unique et nous donnerons ensuite des algorithmes efficaces pour trouver cette solution.

Rappelons d'abord que le fait qu'un graphe soit planaire est une propriété purement topologique et qu'il existe un théorème du à

Kuratowski **[Bondy & Murty]** qui caractérise les graphes planaires de la manière suivante :

Un graphe est planaire si et seulement si il ne contient pas les deux graphes non-planaires K_5 et $K_{3,3}$ soit le graphe complet à cinq sommets et le graphe biparti complet à trois et trois sommets.

Contenir signifie ici contenir le graphe ou un graphe obtenu en ajoutant des sommets sur les arêtes.

figure III-2 K_5 et $K_{3,3}$

K5 **K3,3**

III-2-1 Connexité

Nous allons commencer par quelques définitions concernant la connexité des graphes :

- Un graphe est connexe si et seulement si pour toute paire de sommets il existe un chemin les joignant.

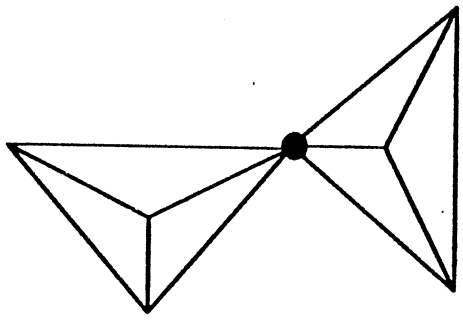
- Un graphe est deux-connexe si et seulement si on ne peut pas le rendre non connexe en enlevant un sommet. Ou encore si pour toute paire de sommets il existe deux chemins disjoints les joignant.

Quand un graphe n'est pas deux-connexe on appelle point d'articulation un sommet qu'il suffit de retirer pour déconnecter le graphe.

- Un graphe est trois-connexe si et seulement si on ne peut pas le rendre non connexe en enlevant deux sommets. Ou si pour toute paire de sommets il existe trois chemins disjoints les joignant.

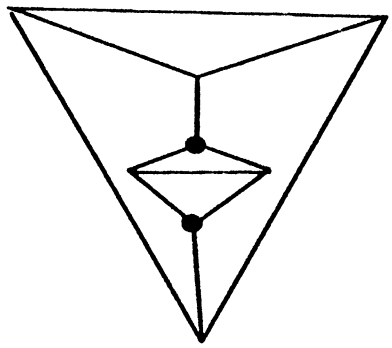
Pour un graphe non trois-connexe, on appelle paire déconnectante un couple de sommets qui déconnecte le graphe quand on le supprime.

figure III-3 connexité



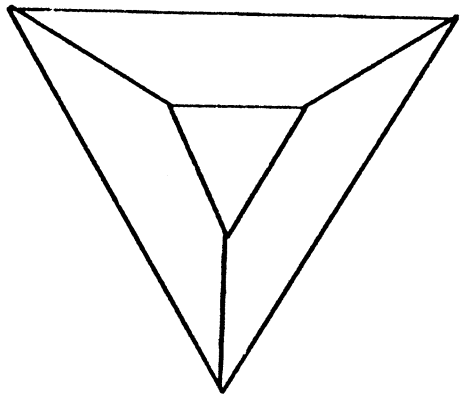
graphe connexe

point d'articulation



graphe deux-connexe

paire deconnectante



graphe trois-connexe

III-2-2 Unicité du solide associé à un graphe planaire

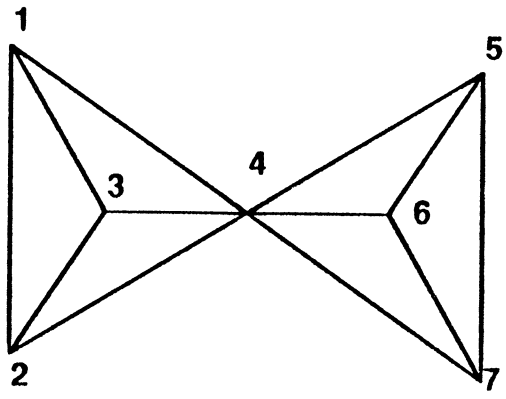
Si le problème que nous étudions n'est pas résolu dans le cas général il se trouve qu'un critère simple d'unicité du solide existe pour les graphes

planaires (pour un solide de genre 0, c'est-à-dire sans trous ni poignées).

L'ambiguïté due à la connexité et à la deux-connexité est assez facile à comprendre. Pour un graphe comprenant un point d'articulation on voit immédiatement que toute les rotations sont possibles autour de ce point et que le sommet ne sera jamais 2-manifold, car l'ensemble des faces incidentes au sommet d'articulation sera divisé en groupes non connexes.

En fait donc un graphe simplement connexe n'admet pas d'interprétation solide correcte, on peut pourtant lui appliquer l'algorithme d'Edmonds. Faisons le sur un exemple :

figure III-4 faces d'un graphe 1-connexe



permutations aux sommets

1 : 2 4 3
2 : 1 3 4
3 : 1 4 2
4 : 1 5 6 7 2 3
5 : 4 7 6
6 : 4 5 7
7 : 4 6 5

On obtient alors les faces suivantes :

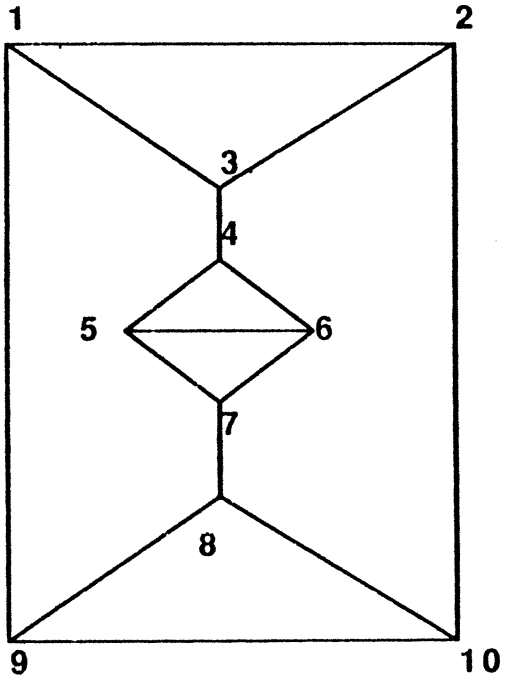
123, 324, 546, 567, 764, 145742

Mais on s'apercoit que la dernière face 145742 n'est pas correcte car le sommet 4 apparait deux fois dans sa frontière, ce qui implique que l'intérieur de la face n'est pas connexe. Il nous faut donc vérifier l'algorithme d'Edmonds pour éliminer de telles faces qui contiennent plusieurs fois le même sommet.

Les graphes deux-connexes peuvent être des solides corrects, même des polyèdres mais non convexes. On trouvera en effet une démonstration dans **[Grünbaum]** du théorème de Steinitz, qui énonce que les graphes planaires trois-connexes sont les squelettes des polytopes à trois dimensions, les polytopes sont les polyèdres convexes, donc les polyèdres convexes et tous les solides pouvant être continûment déformés pour obtenir des polyèdre convexes on des graphes associés planaires trois-connexes.

On voit sur la figure suivante un graphe deux-connexe qui présente un polyèdre non-convexe, et l'on s'apercoit qu'il existe une ambiguïté sur les faces mettant en jeu la paire deconnectante. Cette ambiguïté ne peut être levée que par des critères géométriques.

figure III-5 graphe 2-connexe



Ambiguïté :

On peut remplacer les faces 9.8.7.5.4.3.1 et 2.3.4.6.7.8.10

par les faces 9.8.7.6.4.3.1 et 2.3.4.5.7.8.10

Ici encore l'ambiguïté ne peut être levée que par des informations géométriques.

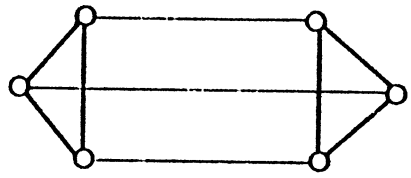
Finalement les seuls graphes dont on soit surs qu'il n'admettent qu'une seule interprétation solide sont les graphes planaires trois-connexes. Remarquons cependant qu'il existera toujours deux possibilités d'orienter les faces du solide, qui correspondent aux deux orientations directe et indirecte (voir deuxième partie), le choix direct-indirect ne peut être fait que sur des critères géométriques pour déterminer quelle orientation correspond à une partie finie de l'espace.

III-2-3 Algorithme de recherche des faces d'un graphe planaire

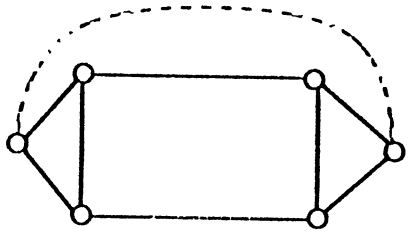
Il existe plusieurs algorithmes de reconnaissance de graphes planaires qui énumèrent les faces (**[Tarjan, Rubin, Demoucron]**). Ces algorithmes savent répondre efficacement aux trois questions suivantes : le graphe est-il planaire, est-il trois-connexe, quelles sont les faces. Nous allons présenter celui de Demoucron, Pertuiset et Malgrange **[Demoucron & all.]** on le trouve aussi chez Rubin **[Rubin]** et Hanrahan **[Hanrahan]**, nous suivrons la présentation qu'en donnent Bondy et Murty **[Bondy & Murty]**

Soit **G** un graphe planaire et **H** un sous-graphe planaire de **G**, soit **H'** une immersion de **H** dans le plan, nous dirons que **H'** est **G-admissible** si il existe une immersion **G'** de **G** dans le plan qui contienne **H'**. La figure **III-6** montre deux immersions d'un sous-graphe dont l'une est admissible et l'autre ne l'est pas.

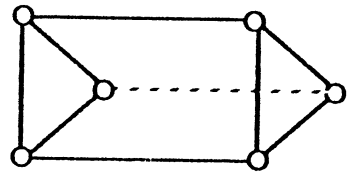
figure III-6 sous-graphe admissible



G



G-admissible

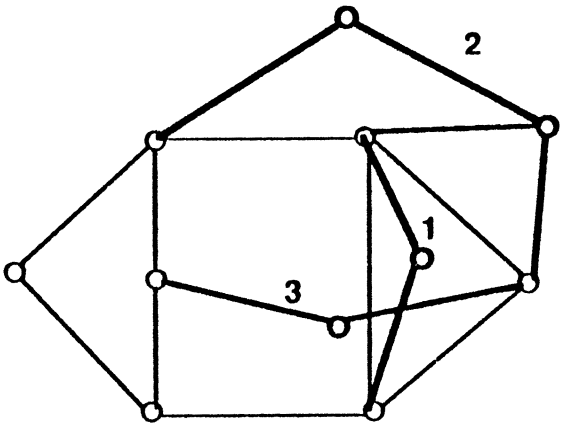


non G-admissible

On appelle pièces de **H** par rapport à **G** les composantes connexes du graphe G-H. Nous dirons qu'une pièce **P** est dessinable dans une face **f** de **H'** si tous les sommets communs à P et H sont sur la frontière de **f**. On note **F(P,H')** l'ensemble des faces de **H'** dans lesquelles **P** est dessinable.

figure III-7 pièces

Le sous-graphe est en traits fins, la pièce 1 peut se dessiner dans trois faces, la pièce 2 dans une seule face (la face extérieure) et la pièce 3 dans aucune face (attention, cela ne signifie pas que le graphe n'est pas planaire mais que le sous-graphe n'est pas admissible, on peut le redessiner pour le rendre admissible).



Il est immédiat que si **H'** est admissible alors **F(P,H')** est non vide pour toutes les pièces de **H**, il suffit de regarder le dessin de **G'** pour y constater que chaque pièce est dessinée dans une face de **H'**, sinon il y aurait des intersections d'arêtes.

L'algorithme de Demoucron, Pertuiset et Malgrange procède en construisant une suite de sous-graphes G_1, G_2, ..., G_n de **G**, planaires avec leurs immersions G'_1, G'_2, ..., G'_n qui soient G-admissible. G_1 est un cycle de **G**, $G_n = G$, et chacun des graphes est obtenu à partir du précédent en choisissant une chaîne et en la plaçant dans une des faces partageant celle-ci en deux faces.

Pour que l'algorithme soit correct il faut que cette opération partant d'un sous-graphe G-admissible G_i construise un sous-graphe G_{i+1} qui soit aussi G-admissible. Pour cela on procède de la manière suivante :

- On établit le liste des pièces de G_i par rapport à **G**, pour chacune de ces pièce on considère $F(P,G'_i)$ et on cherche les conditions suivantes dans l'ordre :

- Si il existe une pièce **P** qui n'a qu'un seul sommet commun avec G_i alors **G** n'est pas 2-connexe, Si il existe une pièce P qui n'a que deux sommets communs avec G_i alors **G** n'est pas 3-connexe.

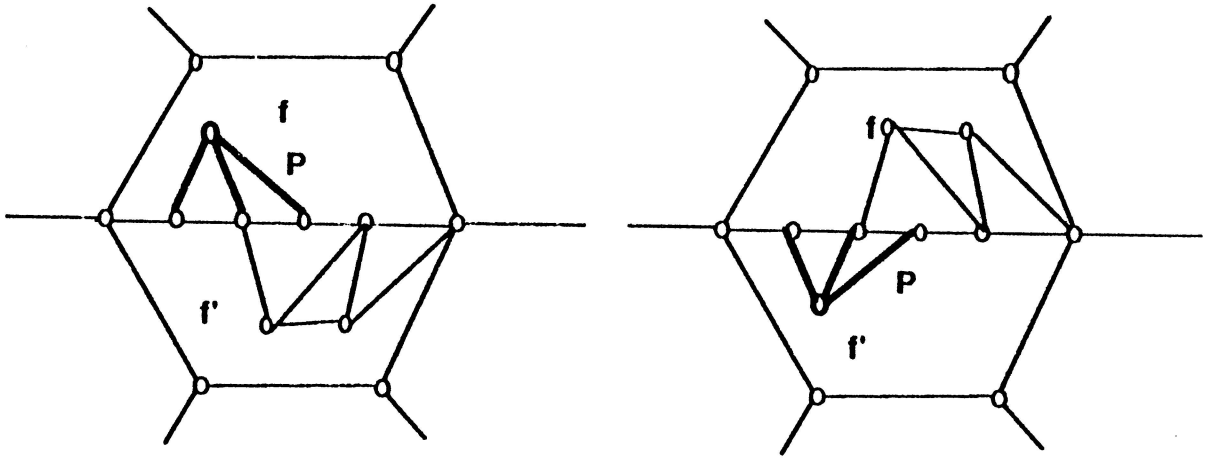
- Si il existe une pièce **P** telle que $F(P,G'_i)$ soit vide alors le graphe **G** n'est pas planaire, comme nous l'avons vu plus haut.

- Si il existe une pièce **P** telle que $F(P,G'_i)$ ne contienne qu'une seule face, alors dans **G'** cette pièce ne peut être que dans cette face, on choisit donc une chaîne dans la pièce **P** et on la place dans la face ce qui donne G'_{i+1} qui est admissible.

- Si pour toute les pièces **P**, $F(P,G'_i)$ contient au moins deux faces, alors on choisit n'importe quelle pièce **P**, on y prend une chaîne que l'on place dans n'importe quelle face de $F(P,G'_i)$. G'_{i+1} ainsi obtenu est admissible, en effet toutes les pièces peuvent être dessinées dans au moins deux faces, si jamais dans **G'** la pièce **P** n'est pas dessinée dans la face que nous avons choisie nous pouvons toujours construire un autre **G'** en changeant les pièces de faces pour que G'_{i+1} devienne admissible (figure III-8).

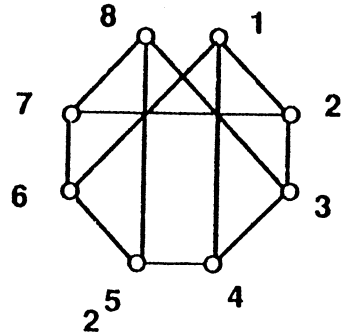
figure III-8 permutation des pièces pour rendre admissible

La pièce P peut se dessiner indifféremment dans f et f'

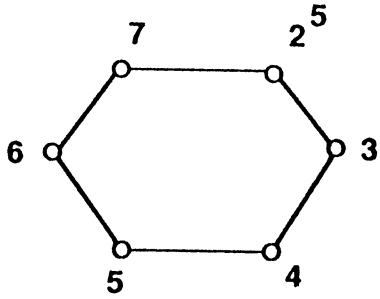


Cet algorithme est efficace puisque le nombre d'étapes est égal au nombre de faces et chaque étape prend un temps constant si on prend soin de conserver l'information sur les pièces en ne recalculant que ce qui est modifié.

La figure ci-contre (**figure III-9**) montre le déroulement de l'algorithme sur un graphe.

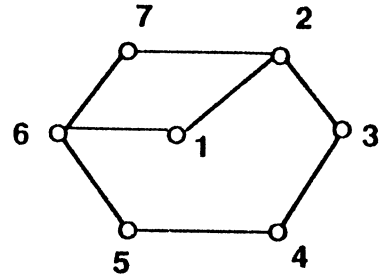


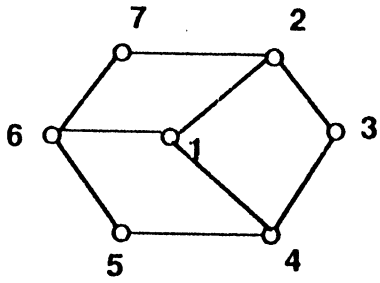
Graphe de départ (cube)



Cycle initial 2 faces
deux pièces : 16,14,12 et 87,85,83
on place la chaine 216

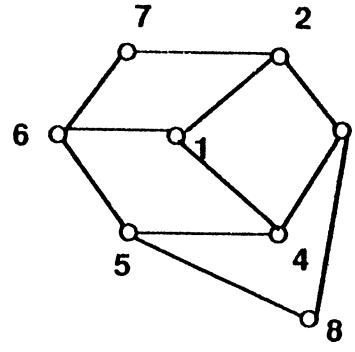
trois faces
deux pièces : 87,85,83 et 14
14 ne peut etre placée que
dans une seule face

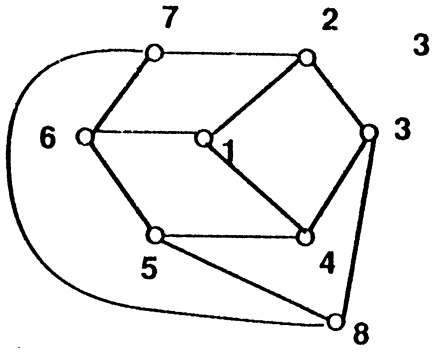




quatre faces
une pièce 87,85,83 qui ne peut etre
placée que dans la face extérieure
on place la chaine 385

5 faces
reste une pièce 87
qui ne peut etre placée que dans
une seule face





Résultat final 6 faces de quatre arêtes

III-3 Remarques sur l'approche topologique

L'approche topologique parait séduisante car elle permet de résoudre un problème a priori géométrique par de moyens uniquement combinatoires qui sont beaucoup plus efficaces. Cependant nous n'avons pas pu retenir cette approche à cause des nombreuses limitations suivantes :

- Elle est limitée au problème du fil-de-fer, on ne voit pas comment traiter le problème des projections par cette méthode, on verra dans la quatrième partie que les fil-de-fers intermédiaires construits pour résoudre le problème des projections n'ont pas du tout la structure requise pour appliquer la méthode topologique.

- Elle est limitée aux graphes planaire trois-connexes, en effet dans les autres cas les ambiguïtés sont inacceptables et il faut ajouter des calculs géométriques pour les lever, ce n'est que quand la solution est unique que l'on pourra oser la présenter sans risquer de proposer un résultat manifestement faux. Hors ces graphes planaires trois-connexe sont relativement peu fréquents dans les problèmes rencontrés.

- Elle est limitée aux polyèdres, ceci peut paraître paradoxal puisque l'idée de l'approche topologique est justement de s'affranchir des formes géométriques, cependant l'information que nous cherchons finalement est bien une information géométrique et ce n'est que dans le cas des polyèdres que nous n'aurons pas à réexaminer chaque face pour retrouver l'équation géométrique de la surface sous-jacente.

Finalement nous n'avons pas adoptée cette approche, cependant elle peut être intéressante pour reconstruire de volumineux modèles polyédriques. En effet les approximations polyédriques de solides à surfaces courbes contiennent un grand nombre de faces et occupe de grands espaces mémoire. On pourrait imaginer pour les échanges de données ou les stockages sur disques de ne garder que le fil-de-fer, sachant qu'il existe un algorithme extrêmement rapide pour reconstituer l'ensemble des faces.

IV Reconstruction de solides

Nous présentons maintenant un algorithme pour retrouver tous les solides correspondant à un fil-de-fer ou à un ensemble de vues orthogonales. Nous allons d'abord décrire l'architecture générale de cet algorithme avant de détailler les différentes étapes.

IV-1 Principe de l'algorithme

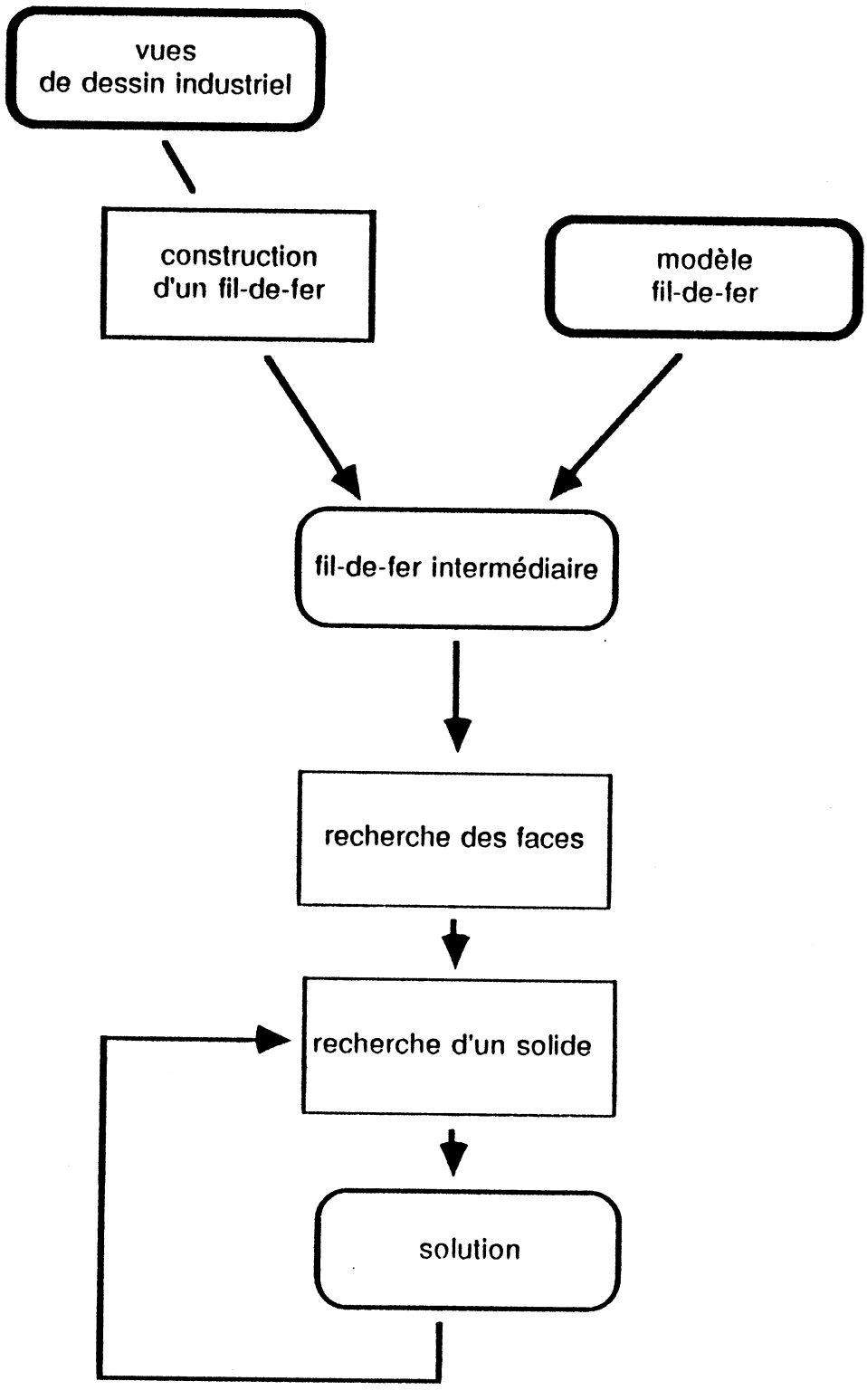
Comme on le voit sur le diagramme de la page suivante l'algorithme accepte deux types de données : un fil-de-fer composé de sommets et d'arêtes ou un ensemble de deux ou trois vues orthogonales du type dessin industriel. Dans ce dernier cas l'algorithme commence par construire le fil-de-fer associé aux vues, on verra par la suite que ce fil-de-fer contient plus d'arêtes et de sommets que le solide recherché. Ensuite, à partir de ce fil-de-fer l'algorithme cherche à construire des surfaces et des faces, et à retrouver les arêtes de tangences qui peuvent manquer sur un fil-de-fer issu de vues.

principe d'exhaustivité:

L'algorithme procède par étapes successives en essayant à chaque étape de construire des objets : d'abord les sommets, puis les arêtes puis les surfaces et les faces. Dans chaque cas la technique est similaire, nous cherchons une condition nécessaire que doivent vérifier les objets puis nous énumérons tous les objets qui satisfont cette condition. On comprend dans ce cas que l'on va bien reconstruire tous les objets (ce qui est indispensable), mais que l'on va aussi en produire de surnuméraires. Il est cependant impossible de donner des conditions nécessaires et suffisantes sauf à la dernière étape, c'est à dire en construisant les solides qui correspondent aux données initiales.

Nous allons commencer par expliquer la construction du fil-de-fer associé aux vues. Cette construction se passe en quatre étapes qui seront détaillées.

- Vérification et préparation des données, recherche des sommets.
- Recherche des arêtes, recherche des arêtes silhouette.


vues
de dessin industriel
construction
d'un fil-de-fer
modèle
fil-de-fer
fil-de-fer intermédiaire
recherche des faces
recherche d'un solide
solution

IV-2 Construction d'un fil-de-fer associé aux vues

IV-2-1 Vérification et préparation des données

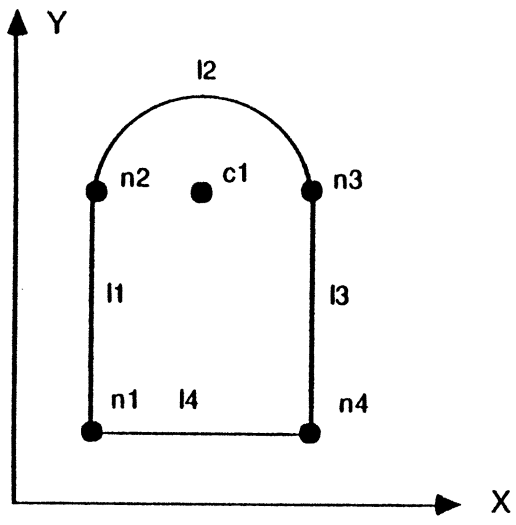
Les données sont deux ou trois vues de dessin industriel. Ils existe trois types de vues qui sont :

- 1 . vue Y-Z projection parallèle à l'axe X. vue de dessus ou de dessous.

- 2 . vue X-Z projection parallèle à l'axe Y. vue de gauche ou de droite.

- 1 . vue X-Y projection parallèle à l'axe Z. vue de face ou de derrière.

figure IV-1 structure d'une vue



Noeuds
1 - (1, 1, 0)
2 - (1, 4, 0)
3 - (4, 4, 0)
4 - (4, 1, 0)

Lignes
1 - droite - 1 , 2
2 - cercle 1 - 2 , 3
3 - droite - 3 , 4
4 - droite - 4 , 1

Centres
1 - (2.5 , 4 , 0)

Cercles
1 - centre 1
rayon 1.5
sens négatif

Elles se présentent comme des ensembles d'arêtes et de sommets en deux dimensions. Nous appelerons **noeuds** les sommets en deux dimensions et **lignes** les arêtes. Les noeuds sont des points **(x,y,0)**, **(x,0,z)** ou **(0,y,z)**, les lignes sont décrites par leurs extrémités, deux noeuds, et par une courbe support de la ligne. Actuellement ces courbes sont des droites implicites ou des cercles, on ajoute alors le centre, le rayon et le sens de rotation; la ligne est l'arc de cercle compris entre le premier et le deuxième noeud dans le sens de rotation.

Les noeuds et les lignes sont enregistrés dans deux tables et reçoivent chacun un indice, on vérifie de plus qu'il n'y a pas de noeuds confondus ou de lignes confondues.

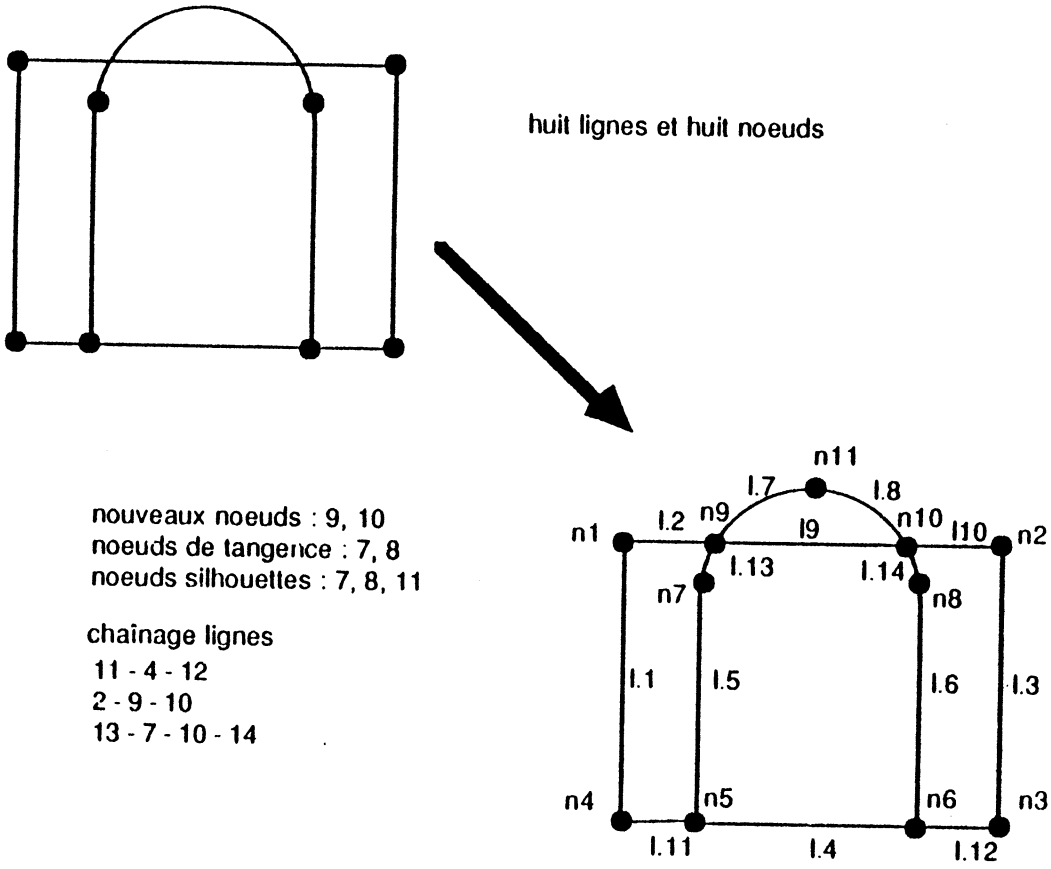
De plus il faut comparer les lignes entre elles pour déterminer si elles ne s'intersectent pas, dans ce cas on crée un nouveau noeud au point d'intersection et on sépare les lignes coupées en plusieurs nouvelles lignes.

On construit encore certains noeuds particuliers : noeuds de tangence, ce sont les noeuds où deux lignes ont un raccord tangent du premier ordre, noeuds silhouettes, ce sont les noeuds où les courbes (arcs de cercles) présentent des tangentes parallèles aux axes de projections (axes de référence).

Enfin pour accélérer les calculs suivants nous introduisons un chaînage entre les lignes qui sont supportées par la même courbe et qui ont un sommet commun. Pour que ce chaînage soit correct il faut que toutes les lignes supportées par une même courbe soient orientées dans le même sens. Pour les cercles c'est déjà assuré. Pour les droites nous orientons tous les segments dans le sens de la première coordonnée croissante puis de la deuxième coordonnée si les premières sont égales.

L'ensemble de ces opérations nécessite un coût en calcul de l'ordre de n^2 opérations élémentaires (intersections, comparaisons) où **n** est le nombre de lignes.

figure IV-2 traitement d'une vue



IV-2-2 Construction des sommets

Suivant leurs projections il faut distinguer plusieurs types de sommets. Les projections d'un sommet sur des vues sont des points qui doivent naturellement partager certaines coordonnées. Si le sommet a pour coordonnées **(x, y, z)** alors les projections sont **(x, y)**, **(x, z)** et **(y, z)**. Nous allons donner les conditions pour construire les sommets à partir de trois vues, dans le cas où il n'y a que deux vues il suffit d'affaiblir ces conditions.

- Sommet " franc "

Ce sommet se trouve à l'intersection d'au moins trois arêtes non coplanaires. Sa projection sur chaque vue est un noeud. Pour construire

tous les sommets francs on parcourt tous les noeuds d'une première vue, pour chacun on parcourt tous les noeuds d'une deuxième vue pour découvrir ceux qui ont une coordonnée en commun avec le premier, dans ce cas on crée un sommet. Si il y a une troisième vue on vérifie que le sommet se projette bien sur un noeud de cette vue.

- Sommet " silhouette "

L'une des arêtes aboutissant à ce sommet est une arête silhouette. Ce sommet se projette sur un noeud, un noeud silhouette et un point intérieur à une ligne. De plus la tangente au noeud silhouette indique la direction de projection sur la vue où se trouve le noeud. Pour trouver ces sommets on explore dans chaque vue l'ensemble des noeuds silhouettes et pour chacun on vérifie sur les autres vues les conditions nécessaires. Il nous faut être capable d'intersecter une ligne de coordonnée constante avec une ligne pour découvrir les points intérieurs aux lignes.

- Sommet " tangence "

L'une des arêtes est une arête de tangence. Tous les sommets de ce type qui ne sont pas déjà découvert dans les deux étapes précédentes vérifient la même propriété. Une des projections est un noeud de tangence, les deux autres projections sont des points intérieurs à des lignes. La construction de ces sommets est similaire à celle des sommets silhouettes en partant de tous les noeuds de tangence.

Il reste deux cas de sommets beaucoup moins répandus, mais nécessaires pour la généralité :

- sommet de cône

Dans ce cas particulier l'une des projections est le centre d'un arc de cercle sur une vue. Nous ne considérons là que des cônes d'axe parallèle aux axes de projection. En fait nous pourrions considérer les centres d'arcs de cercle comme des cercles de rayon nul. Pour trouver de tels sommets nous introduisons des noeuds fictifs au centre des arcs de cercles. Les sommets de cônes sont alors construits avec les sommets francs.

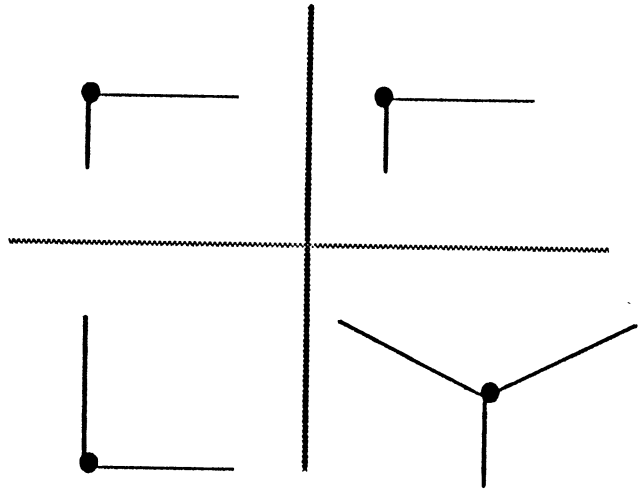
- Sommet double-silhouette

Il s'agit du cas particulier d'un point extrême des contours apparents. Sur ce sommet ne se rencontrent que des arêtes silhouettes. Il en existe deux types, sommets elliptiques (sphère par exemple) avec deux courbures de même signe, ou hyperboliques (sur le tore par exemple) avec des courbures de signes opposés. Ces sommets se projetent sur deux noeuds silhouettes tels que la tangente en chacun de ces noeuds est parallèle à la direction de projection vers l'autre noeud. La troisième projection est un point hors de tout noeud ou ligne. Remarquons que les sommets elliptiques peuvent se projeter sur un centre de cercle et donc être identifiés avec les sommets de cône. Mais ce n'est pas le cas des sommets hyperboliques.

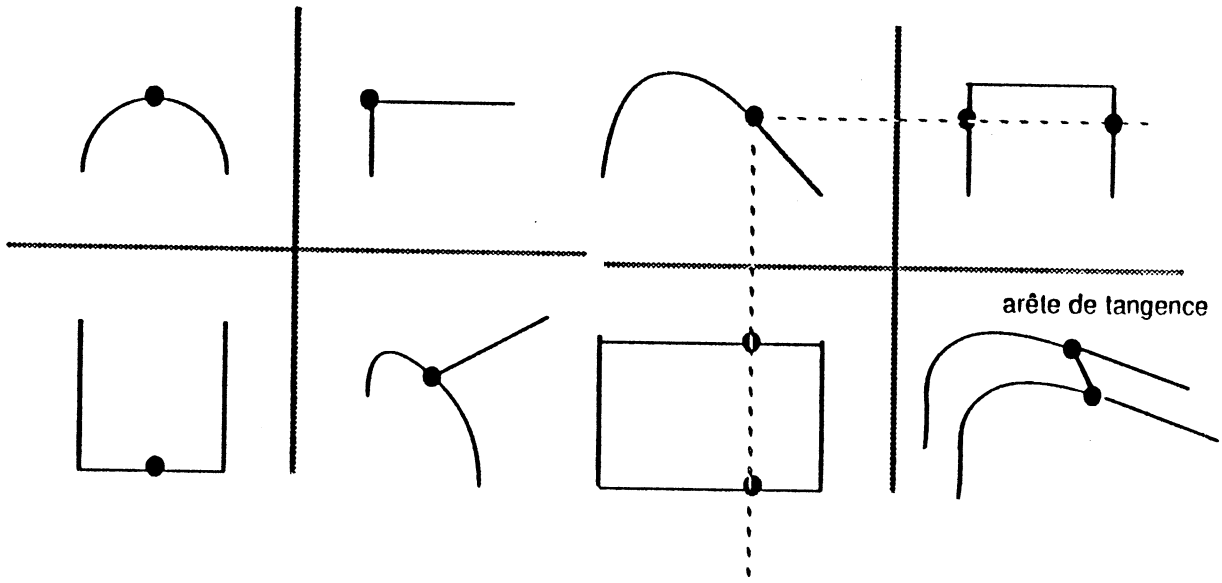
Toutes ces conditions peuvent être considérées comme des heuristiques car elles ne donnent pas en effet tous les sommets dans tous les cas. De plus elles peuvent ajouter des pseudo-sommets qui ne font partie d'aucun solide compatible avec les vues. Les sommets non découverts correspondent aux solides qui ne sont pas encore identifiés par l'algorithme, nous décrirons plus tard la classe des solides reconnus et non-reconnus.

figure IV-3 les différents sommets

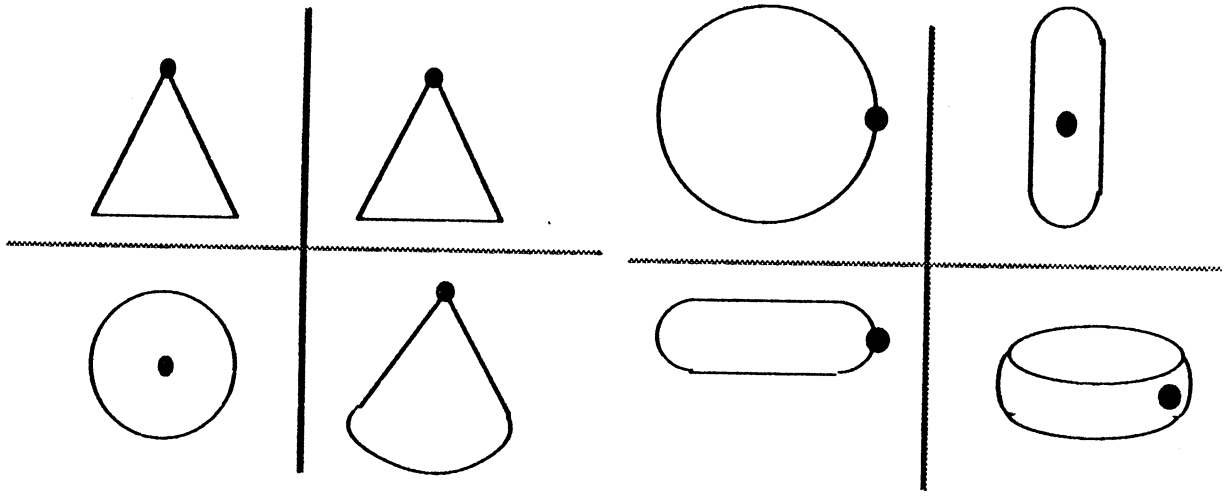
sommet franc



sommet silhouette - sommet tangence



sommet de cône - sommet double-silhouette



Remarquons encore que notre algorithme construit une table de sommets en attribuant à chacun un indice entier. Il est très important pour le bon fonctionnement des procédures topologiques qu'il n'y aie pas deux sommets confondus, c'est-à-dire avec les mêmes coordonnées et des indices différents. Pour cela, à chaque nouveau sommet créé nous vérifions s'il n'appartient pas déjà à la table. En fait certaines considérations dans les algorithmes d'exploration des noeuds permettent de limiter le coût de cette recherche. Deux sommets sont considérés comme identiques si les différences de leurs coordonnées sont inférieures en valeur absolue à un paramètre epsilon judicieusement choisi, en général 10^{-5} fois la plus grande dimension du modèle. Ce paramètre sert aussi à décider si deux noeuds ont une coordonnée identique et sont donc candidats à former un sommet, dans ce dernier cas il est possible de remonter le paramètre epsilon pour tenir compte d'un léger désalignement des vues, on peut monter jusqu'à la plus petite distance caractéristique entre noeuds sur une vue. Alors l'algorithme adoptera comme coordonnée du sommet la moyenne des deux coordonnées des noeuds.

Avec chaque sommet de la table nous enregistrons ses projections sur les deux ou trois vues. Cette projection est un entier qui peut être :

- positif : la projection est un noeud et l'entier est l'indice de ce noeud dans la table des noeuds. On peut déterminer à partir de cet indice si il s'agit d'un noeud silhouette ou tangence ou encore d'un centre de cercle.

- négatif : la projection est un point à l'intérieur d'une ligne. La valeur absolue de l'entier est alors l'indice de cette ligne dans la table des lignes.

- nul : la projection est un point hors des noeuds et lignes, ou la vue n'existe pas.

IV-2-3 construction des arêtes.

Examinons maintenant les sommets que nous avons construits et essayons de les joindre par des arêtes. Soit une paire de sommets S1, S2 quelles sont les conditions pour qu'il soient reliés par une arête ?. Soient N1 et N2 les projections des deux sommets sur une vue, N1 et N2 sont des entiers décrivant les projections, tels qu'il ont été introduits au paragraphe précédent.

En considérant les différentes possibilités de projeter une arête nous allons imaginer une procédure **aligné(N1, N2)** qui nous dira si sur une vue une arête peut exister entre deux sommets.

Les sommets se projettent sur deux points, ces deux points peuvent être confondus, il y a alors possibilité d'une arête rectiligne perpendiculaire à la vue. Sinon les points peuvent être reliés par une ligne ou par une chaîne de lignes, mais dans ce dernier cas toutes les lignes de la chaîne doivent être supportées par la même courbe. C'est dans le but d'accélérer cette procédure que nous avons chaînées entre elles les lignes supportées par la même courbe. Observons que les indices N1 et N2 peuvent êtres positifs ou négatifs. S'ils sont égaux alors soit ils sont positifs et il s'agit du même sommet, soit ils sont négatifs et appartiennent à la même ligne.

Si ils sont différents il faut lancer une recherche de chaîne de N1 vers N2, puis éventuellement de N2 vers N1 . Le départ de la recherche se fait sur toutes les lignes quittant N1 si c'est un noeud, sur la ligne si c'est une ligne. L'exploration poursuit par le chaînage et s'arrête quand l'extrémité de la ligne est N2 si N2 est positif (noeud) ou si la ligne est égale à - N2 si N2 est négatif.

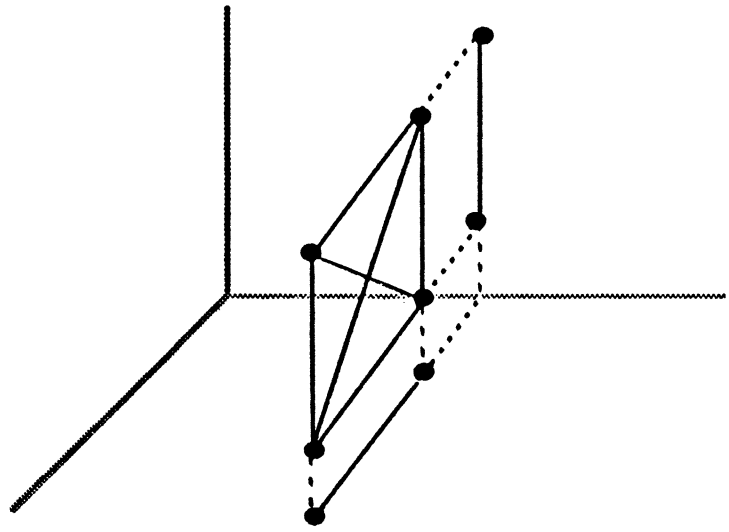
```
procedure aligné ( N1, N2 : integer) : boolean;
{ cette procédure détèrmine si il existe une projection d'arête entre les projections de
sommets N1 et N2 }
aligné := faux ; { valeur par défaut }
si (N1 = 0) ou (N2 = 0) alors retour;  {projection hors de toute ligne }
si (N1 = N2 ) alors
début
aligné := vrai;
retour
fin;                    { même noeud ou même ligne }
{ recherche d'une chaine de N1  vers N2 }
pour N  = toutes les lignes quittant N1 si N1 > 0
N  = - N1                              si N1 < 0
faire
    { recherche d'une chaine démarrant par N }
    tant que ( N  <> 0 ) faire
    début
    si      ( ( N2 > 0 )  et  ( extremité (N)  = N2 ) )
    ou
    ( ( N2 < 0 )  et  ( N = - N2 ) )  alors
    début
            aligné := vrai;
            retour
    fin;    { fin de la chaine }
    N  = prolonge(N) ;   { on considère la ligne suivante dans la chaîne }
fin;

{ Même recherche pour une chaîne partant de N2 vers N1 }
fin;
```

Nous sommes maintenant capables de décider si une arête relie deux sommets, il suffit de projeter ces sommets sur chaque vue et d'utiliser la procédure précédente. Pour déterminer la courbe qui supporte l'arête en trois dimensions il nous faut intersecter les (deux ou trois) surfaces obtenues en translatant les courbes en deux dimensions qui supportent les lignes. Dans la pratique nous nous sommes contentés des droites et des arcs de cercles situés dans des plans parallèles aux vues, cependant

il est envisageable d'introduire des courbes plus complexes, particulièrement les courbes d'intersections comme les coniques. Un cas particulier n'entre pas dans ce schéma, c'est celui de deux vues où les courbes projections sont des droites parallèles aux axes de projections . Dans ce cas en effet les deux surfaces sont confondues en un même plan parallèle à la troisième vue (absente), et leur intersection n'est évidement pas une courbe mais ce même plan. L'arête peut alors être théoriquement n'importe quelle courbe tracée sur le plan et restant dans le rectangle défini par les deux segments entre les quatre sommets. En particulier pour les arêtes rectilignes, les quatre côtés du rectangle et ses deux diagonales (figure III-5). Nous avons choisi dans ce cas, pour éviter une explosion combinatoire du nombre d'arêtes, de ne garder que les quatre côtés, c'est-à-dire les arêtes parallèles aux axes de projections (ou axes de référence). Ce choix implique qu'une vue peut être omise sur les trois seulement s'il n'existe pas d'arête non parallèle aux axes qui soit contenue dans un plan parallèle à la vue et ne soit donc détectable que sur cette vue. Il est possible d'introduire les arêtes diagonales mais on ajoute alors beaucoup d'arêtes et de points d'intersections entre arêtes qui complexifient énormément le calcul pour un gain pas toujours probant, on peut consulter pour cela l'article de Markowsky et Wesley.

figure IV-5 deux lignes parallèles aux axes donnant six arêtes



Nous pourrions dès maintenant proposer un algorithme de recherche des arêtes qui énumère toutes les paires de sommets en cherchant pour chacune si une arête se trouve entre les deux sommets. Cependant cette approche brutale présente deux défauts majeurs :

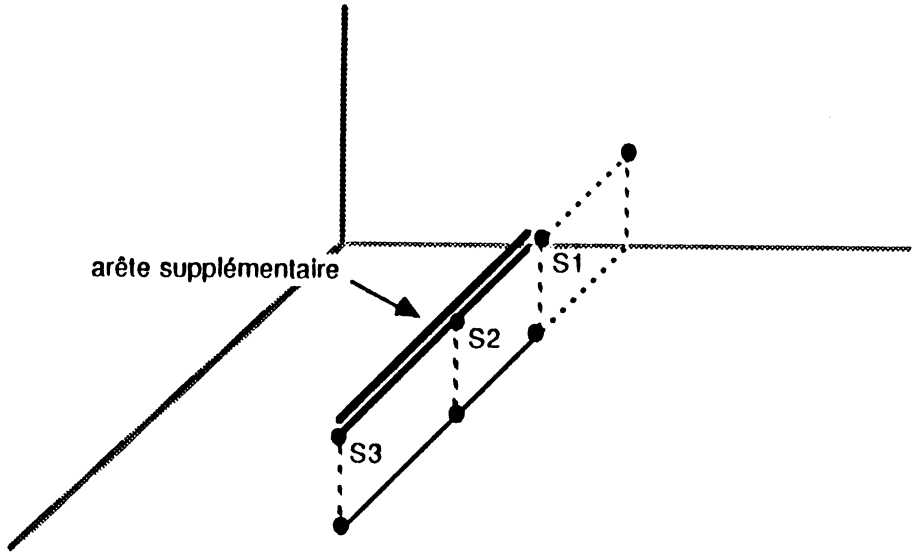
- Complexité

L'énumération des paires de sommets coûte **n (n-1) / 2** où **n** est le nombre de sommets. De plus dans la plupart des cas il n'y aura pas d'arêtes, si on compte de l'ordre de 3 arêtes par sommets il y a **3n / 2** arêtes. Or justement pour chaque paire de sommets il faut appeler deux ou trois fois la procédure aligné qui est particulièrement inefficace quand il n'y a pas d'arête puisqu'elle explore complètement toutes les chaînes à partir des noeuds projections des sommets. Il faudrait donc restreindre la recherche en éliminant a priori les paires de sommets qui ne donnent pas d'arêtes.

- Arêtes alignées

Un autre problème plus grave est le suivant. Imaginons trois sommets S1, S2 et S3 alignés sur une même courbe dans cet ordre, si il y a une arête entre S1 et S2 et une arête entre S2 et S3 alors notre algorithme construira automatiquement une arête entre S1 et S3. On se rend facilement compte que pour quatre sommets il y aura non plus une mais trois arêtes supplémentaires, en général **p(p-1) / 2 - p** où **p** est le nombre de sommets alignés. Outre leur prolifération ces arêtes sont indésirables car elles perturbent le déroulement de la suite de l'algorithme; en effet rappelons que nous avons exigé que les arêtes du solide ne puissent s'intersecter hors de leurs sommets, ce qui n'est manifestement pas le cas ici où des arêtes sont incluses les unes dans les autres. Il serait coûteux d'ajouter une procédure de nettoyage qui parcoure les arêtes pour éliminer toutes celles qui contiennent un sommet hors de leurs extrémités. Il nous faut donc repenser notre algorithme pour éviter de générer de telles arêtes.

figure IV-6 arêtes multiples alignées



Le nouvel algorithme que nous proposons pour rechercher les arêtes part de la constatation que de nombreuses arêtes sont parallèles aux axes de projections et se projetent donc sur certaines vues comme de simples noeuds. Convenons d'appeler arêtes axiales de telles arêtes, arêtes semi-axiales les arêtes contenues dans des plans parallèles aux vues et qui se projetent donc sur des lignes parallèles aux axes sur deux vues, et enfin arêtes non-axiales ou arêtes obliques les autres arêtes qui ne se projetent sur aucune ligne parallèle à un axe.

Notre algorithme se sépare en deux phases :

- Recherche des arêtes axiales perpendiculaires à chacune des vues.

- Recherches des arêtes semi-axiales et obliques parallèles aux vues. Les arêtes obliques ne sont cherchées qu'à partir d'une seule vue pour ne pas être répétées.

- Eventuellement, quand il n'y a que deux vues, une troisième phase cherche des arêtes axiales perpendiculaires à la vue manquante.

Recherche des arêtes axiales.

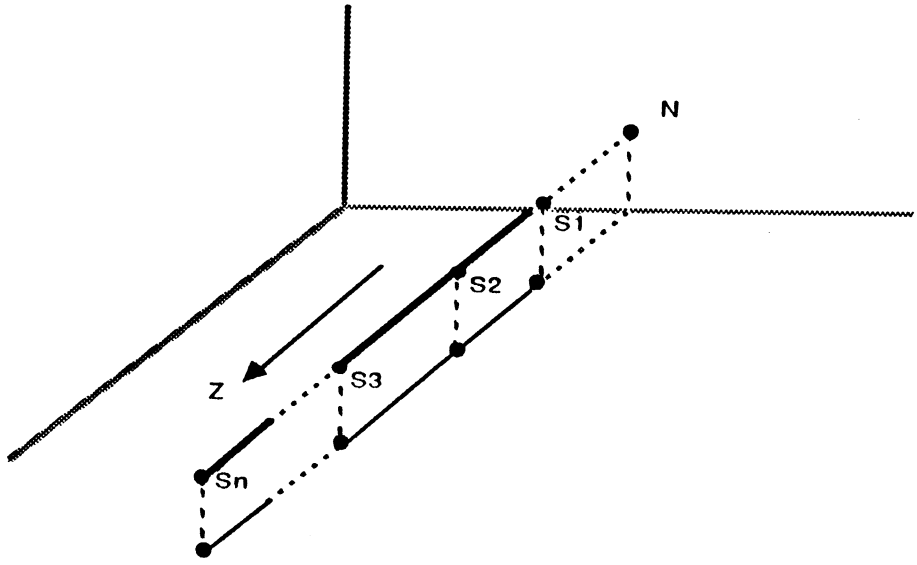
Pour rechercher les arêtes axiales perpendiculaires à une vue donnée on visite tous les noeuds de cette vue et pour chacun de ces noeuds on recherche toute les arêtes perpendiculaires à la vue qui se projetent sur ce noeud. Pour cela on procède de la manière suivante:

Soit N le noeud de la vue considérée, nous construisons l'ensemble S des sommets qui se projetent sur cette vue exactement sur le noeud N. Tous les sommets dans S ont deux coordonnées identiques, celles de N, la troisième coordonnée, appelons la z, mesure l'éloignement du sommet par rapport à N. Nous trions les éléments de S par rapport à z. Soit le résultat de ce tri S = { S1, S2, S3, ... }, nous allons maintenant utiliser la procédure décrite ci-dessous, où N = projection(S,V) signifie que N est la projections du sommet S sur la vue V. Le principe est de faire progresser une paire d'indice i, j en cherchant à établir une arête entre Si et Sj si il y en a une on remonte i = j pour chercher une nouvelle arête à partir de Sj sinon on incrémente j en gardant Si comme origine.{ soit n le nombre d'éléments dans S }

```
{soient V1 et V2 les deux autres vues, V2 = 0 si il n'y a que deux vues }

i = 1; j := i;
répeter
    j := j + 1;
    si      aligné ( projection ( Si, V1 ), projection ( Sj, V1 ) )
    et
    aligné ( projection ( Si, V2 ), projection ( Sj, V2 ) )
    alors
            créer-arête ( Si, Sj );
            i := j;
    fin
jusqu'à  j = n .
```

figure IV-7 arêtes axiales



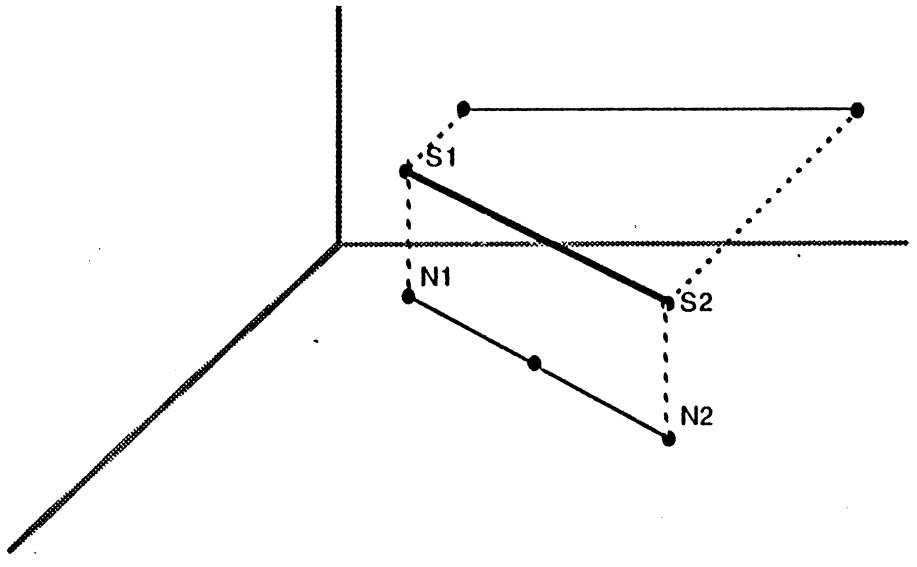
Recherche des arêtes semi-axiales et obliques.

Pour chercher ces arêtes nous ne partons plus des noeuds mais des lignes. C'est à dire que sur chaque vue nous allons examiner chaque ligne non parallèle à un axe, il y en a en fait assez peu en général. Nous considérons alors les deux noeuds N1 et N2 extrémités de cette ligne, à ces noeuds correspondent des ensembles de sommets S1 et S2 , nous allons maintenant chercher à établir des arêtes entre tous les couples de sommets S1 et S2 où S1´ S1 et S2´ S2 . Cette procédure est assez élémentaire mais est légèrement compliquée par le fait que se limiter aux lignes ne suffit pas toujours à trouver toutes les arêtes. En effet, une arête semi-axiale peut se projeter non pas sur une seule ligne mais sur une chaîne de lignes, il nous faut donc étendre notre recherche aux lignes dans le prolongement de la ligne de départ et choisir les couples de noeuds N1 et N2, pas seulement aux deux extrémités d'une ligne mais d'une chaîne de lignes sur la même courbe. Il y a alors risque de créer des arêtes supplémentaires alignées comme nous l'avons déjà expliqué. Pour éviter cela, quand nous construisons l'arête nous enregistrons les noeuds intérieurs aux chaînes qui forment les projections de l'arête, si jamais trois (ou deux s'il n'y a que deux vues) de ces noeuds intérieurs

correspondent à un sommet nous stoppons l'arête car le sommet serait à l'intérieur de l'arête.

Pour les arêtes obliques qui se projettent sur chaque vue sur des lignes non parallèles aux axes il ne faut construire l'arête qu'à partir d'une seule de ces vues pour éviter les répétitions.

figure IV-8 arête semi-axiale



Cas de deux vues

Les procédures précédentes dans le cas de deux vues ne permettent pas de reconstruire les arêtes perpendiculaires à la vue manquante qui se projettent sur les deux vues sur des lignes parallèles aux axes. Pour trouver ces arêtes nous sommes obligés de passer par une procédure plus directe qui consiste à explorer tous les sommets en rassemblant ceux qui se projeteraient au même point sur la vue manquante. Pour chacun de ces ensembles de sommets alignés perpendiculairement à la vue manquante on applique la procédure de la première phase (arêtes axiales). C'est-à-dire que l'on trie les sommets selon la coordonnée perpendiculaire à la vue et l'on applique l'algorithme décrit.

Comme nous avons enregistré avec chaque sommet ses projections sur les vues, nous enregistrons de même avec chaque arête ses projections

qui vont nous servir par la suite. Pour chaque arête et pour chaque vue nous enregistrons un indice entier qui peut être :

- 0 si l'arête se projette sur un noeud, un point, rien du tout, ou si la vue n'existe pas.

- sinon c'est l'indice de la première ligne de la chaîne, pour connaître toutes les lignes de la chaîne sur laquelle se projette l'arête il suffit de suivre les prolongements de cette ligne jusqu'à rencontrer la projection du sommet qui est à l'extrémité de l'arête.

IV-2-4 Construction des arêtes silhouettes

La recherche des arêtes silhouettes à partir des vues est une procédure beaucoup plus heuristique que la précédente, c'est-à-dire que nous ne sommes pas sûrs de trouver toutes les arêtes. Nous partons en effet d'un a priori arbitraire mais très souvent vérifié qui nous simplifie considérablement le travail. Nous supposons que les arêtes silhouettes se trouvent dans des plans parallèles aux plans des vues. La hauteur de ces plans par rapport aux plans de vues nous sera indiquée par les noeuds silhouettes. C'est à partir de ces derniers que nous allons rechercher nos arêtes silhouettes.

Nous partons donc d'un noeud silhouette qui nous définit un plan parallèle à une vue, la vue n'est autre que la vue correspondant à la direction de projection parallèle à la tangente au noeud silhouette. Les sommets silhouettes se projetant sur notre noeud silhouette sont les points de départ des arêtes silhouettes, pour trouver ces dernières nous considérons la projections des sommets sur la vue parallèle et nous cherchons toutes les lignes passant par ces projections. Les arêtes silhouettes correspondent à la translation de ces lignes dans le plan parallèle à la vue.

Remarques importantes :

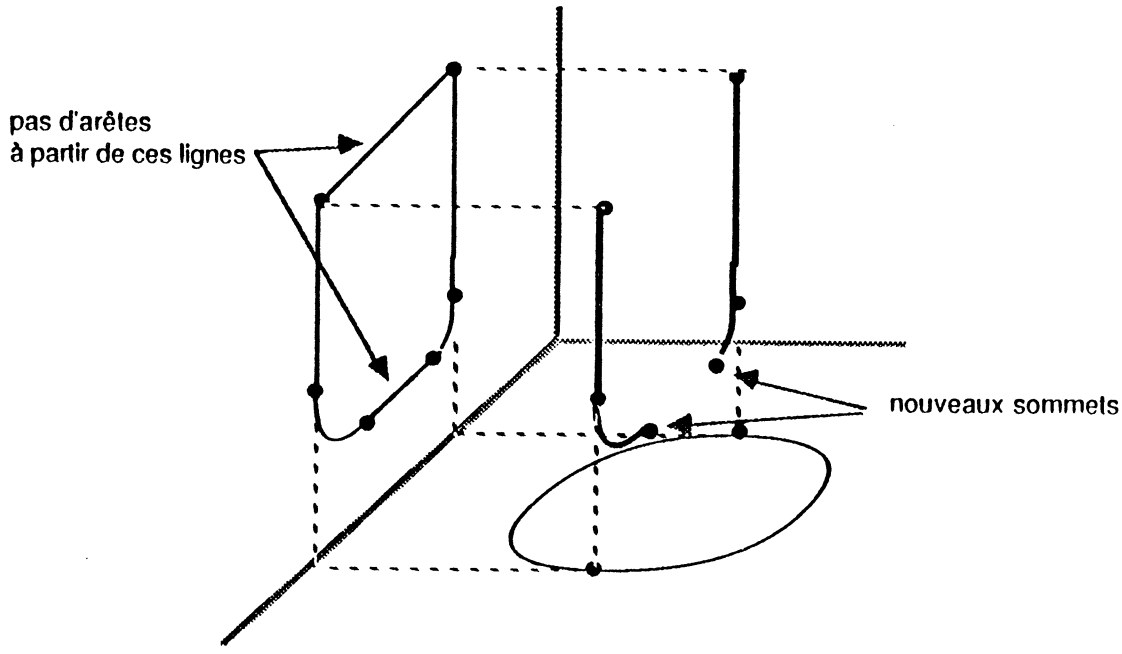
- Nous créons des arêtes silhouettes à partir des lignes passant par les projections des sommets silhouettes mais aussi des lignes dans le prolongement de celles-ci.

- Nous ne prenons pas en considération les lignes parallèles à la vue qui contient le noeud silhouette. Ces lignes conduisent à des arêtes silhouettes incorrectes.

- Certaines arêtes silhouettes ne sont pas identifiées (par exemple l'arête circulaire au sommet d'un tore) .Cependant nous verrons plus tard que ce n'est pas un handicap car ces arêtes ne sont pas nécessaire pour reconstruire les surfaces ou alors ce sont des arêtes de tangence qui seront construites dans une phase ultérieure.

- Certaines lignes peuvent se terminer sur un noeud silhouette, (morceau de tore par exemple). Dans ce cas il peut arriver qu'il manque un sommet, nous ajoutons ce sommet en vérifiant qu'il ne coupe aucune arête (sinon il faut découper l'arête en deux). Ces sommets sont marqués particulièrement car ils sont en général à l'origine des arêtes silhouette-tangence que nous signalons dans la remarque précédente.

figure IV-9 arêtes silhouettes



IV-2-5 Exemples de fil-de-fers associés à des vues

On trouvera en annexe quelques exemples de fil-de-fer associés à des vues. On remarquera les pseudo-sommets et pseudo-arêtes qui sont présents sur le fil-de-fer mais pas sur les solides attendus comme solution du problème. On remarquera aussi les sommets et arêtes fantômes introduits par les alignements fortuits qui viennent découper certaines surfaces.

IV-3 Du fil-de-fer au solide

Nous allons maintenant décrire un algorithme pour trouver tous les solides compatibles avec un fil-de-fer ou un pseudo-fil-de-fer donné (le pseudo-fil-de-fer est le fil-de-fer obtenu à partir de projections et qui peut contenir des pseudo-arêtes et des pseudo-sommets). Cet algorithme se décompose en trois étapes.

- Recherche des surfaces et des arêtes de tangences

On explore le fil-de-fer pour trouver les surfaces susceptibles de porter des faces. Pour chaque surface on établit la liste des arêtes et des sommets qui lui appartiennent. De plus on cherche s'il peut y avoir des arêtes de tangence entre ces surfaces.

- Recherche des faces

Pour chaque surface on traite l'ensemble des arêtes de la surface pour trouver les différentes faces qui partitionnent la surface.

- Recherche des solides

On recherche les assemblages de faces qui correspondent à des solides valides. Dans le cas de vues on vérifie que ces solides restituent bien les vues de départ.

IV-3-1 Recherche des surfaces

Le principe fondamental de cette recherche de surface et de prétendre qu'une surface est définie par deux arêtes partageant un sommet. La première procédure que nous allons utiliser est donc capable à partir de deux telles arêtes de nous donner la ou les surfaces correspondantes. A titre d'exemple voici une table des différentes surfaces que l'on peut obtenir à partir de deux arêtes segments ou arcs de cercles :

- **segment + segment :**
- un plan si les segments ne sont pas colinéaires
- rien sinon.
- **segment + arc de cercle :**
- un cylindre si le segment est perpendiculaire au plan du cercle.
- un plan si le segment est contenu dans le plan du cercle.
- un cône sinon.
- **arc de cercle + arc de cercle :**
- une sphère si les axes des arcs de cercles se coupent.
- un plan si les arcs de cercles sont coplanaires.
- deux tores sinon (en faisant tourner chaque cercle autour de l'axe de l'autre)

Dans le cadre de la représentation fonctionnelle des surfaces et des courbes que nous avons proposée au début de ce rapport il s'agit d'une procédure transparente pour l'algorithme qui à partir de deux arêtes nous renvoie zéro, un ou plusieurs indices de surfaces dans une table des surfaces qui est gérée par le modeleur de surfaces. Si nous ajoutons une procédure pour identifier si deux surfaces sont géométriquement identiques, nous avons un embryon d'algorithme pour reconnaître les surfaces :

Pour chaque sommet, pour chaque paire d'arêtes incidentes au sommet il faut trouver les surfaces définies par les arêtes, chercher si ces surface sont déjà connues et les ajouter si elles ne sont pas connues, puis ajouter les deux arêtes dans la liste d'arêtes de chaque surface.

En fait cet algorithme est prohibitif car il répète inutilement des opérations coûteuses en calcul comme la comparaison des surfaces. Il passe de fait le plus clair de son temps à recréer des objets pour s'apercevoir qu'ils existent déjà après les avoir comparés à tous les autres.

C'est pourquoi l'algorithme que nous avons adopté est un peu différent. Il commence toujours par l'exploration de tous les sommets et pour chaque sommet de toutes les paires d'arêtes incidentes au sommet, seulement pour chaque nouvelle surface créée nous essayons de découvrir immédiatement toutes les arêtes contenues dans cette surface par une exploration du graphe à partir des deux arêtes qui ont défini la surface. Comme nous enregistrons les relations entre arêtes et surfaces nous pouvons déterminer pour chaque paire d'arête si elles n'ont pas déjà été recensées dans une même surface et dans ce cas il est inutile de chercher à créer une nouvelle surface à partir de ces arêtes. Voici une formalisation de cet algorithme.

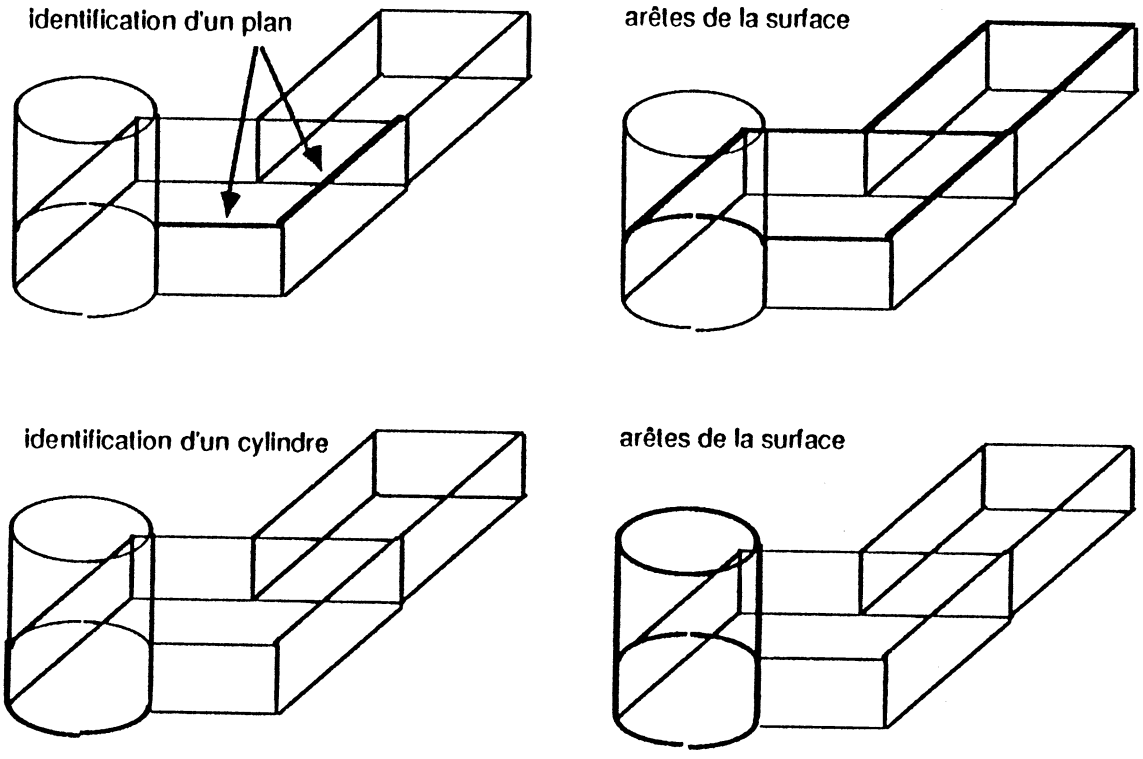
```
{ surf(A) est l'ensemble des surfaces contenant l'arête A }
pour A = toutes les aretes surf(A) := Ø ;
    pour S = tous les sommets
            pour (A1, A2) = toutes les paires d'arêtes incidentes à S
                    si surf(A1) U surf(A2) = Ø alors
                    creer-surfaces(A1, A2) ;
                    { on a ajouté les nouvelles surfaces, peut-etre aucune
                    pour chacune on va chercher les arêtes }
                    pour Σ = chaque nouvelle surface
                    arêtes-surface(S, Σ);
            fin
    fin
fin


procedure arêtes-surface(S, Σ) ;
{ cette procédure recherche toutes les arêtes de la surface Σ qui sont dans le fil-de-
fer en partant du sommet S }
{ nous allons faire une exploration avec une pile des sommets, exploration en
profondeur d'abord, de plus nous marquons les sommets pour retrouver ceux qui
sont explorés }

empiler(S); { on démarre avec le sommet S au sommet de la pile }

tant-que la pile n'est pas vide
    S := dépiler;
    marquer(S);
    pour A = toutes les arêtes partant de S
            si A est sur la surface Σ alors
            surf(A) := surf(A) u {Σ}; { on ajoute la surface à l'arête }
            s := autre sommet de A ; { l'extrémité qui n'est pas S }
            si s n'est pas marqué alors empiler(s);
    fin
fin
```

figure IV-10 recherche de surfaces



Les questions que cet algorithme pose au modeleur surfacique sont finalement les suivantes :

- A partir de deux arêtes ayant un sommet commun donner la liste des surfaces possibles.

- Déterminer si une arête dont on sait qu'une extremité est dans une surface est complètement dans la surface.

Dans ce dernier cas, l'information sur le sommet déjà connu dans la surface peut faire gagner du temps, ainsi pour un plan et une arête arc de cercle, il suffit que l'axe du cercle soit perpendiculaire au plan.

Cet algorithme ainsi présenté est légèrement incorrect et appelle quelques remarques.

- D'abord il peut arriver que deux arêtes (des arcs de cercles par exemple) soit déjà connues dans la même surface (un cylindre par exemple), ce qui les empêche de définir une nouvelle surface (un plan). Pour éviter ce problème il faut introduire quelques modifications de détail.

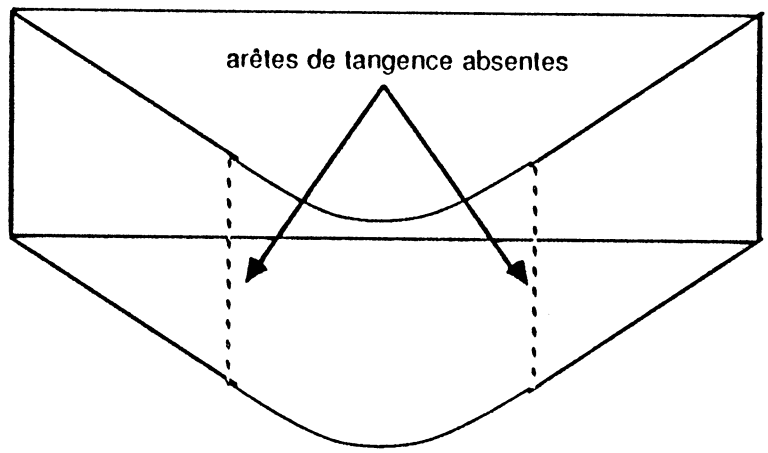
- On remarque aussi que l'algorithme considère comme deux surfaces différentes deux groupes d'arêtes sur une même surface mais sans connexions entre eux. Ceci provient du fait que la procédure qui recherche les arêtes de la surface ne procède que par connexité sur le graphe fil-de-fer. Pour avoir le nombre exact de surfaces il nous faut ajouter une dernière procédure qui rassemble les surfaces identiques. Il est alors nécessaire que notre modeleur surfacique puisse nous dire si deux surfaces sont identiques, ce qui ne pose naturellement aucun problème.

coût : Pour estimer le coût de cet algorithme disons que la boucle sur les sommets est parcourue S fois (S nombre de sommets) , celle sur les paires d'arêtes est constante car il y a environ trois, quatre arêtes par sommet; le coût de comparaison des arêtes pour trouver les surfaces communes est aussi constant car il existe deux ou trois surfaces par arête. Le coût d'exploration d'une surface est proportionnel au nombre d'arêtes qu'elle contient; si nous sommons tous ces coûts nous obtenons la somme des nombres d'arêtes par surface, soit le nombre d'arêtes multiplié par le nombre de surfaces par arête qui vaut deux ou trois. Le coût est donc proportionnel à S*A (S nombre de sommets, A nombre d'arêtes) ce qui est tout à fait raisonnable. Le nombre de surface obtenues peut être très variable mais l'expérience nous donne une moyenne de quatre arêtes par surface, ce qui fait un nombre de surfaces proportionnel au nombre d'arêtes.

IV-3-2 Recherche des arêtes de tangences

Avec l'algorithme de recherche de surfaces décrit au paragraphe précédent nous allons combiner une heuristique pour retrouver les arêtes de tangence absentes du fil-de-fer. Une arête de tangence se trouve entre deux surfaces présentant un raccord avec continuité de normale. Cependant nous ne pouvons pas nous appuyer sur cette définition car il peut se trouver que sans les arêtes de tangence l'une des deux surfaces soit indétectable. Ainsi sur la figure 29, la surface cylindrique ne peut pas être découverte si aucune des deux arêtes de tangence n'est présente dans le modèle.

figure IV-11 surface indétectable

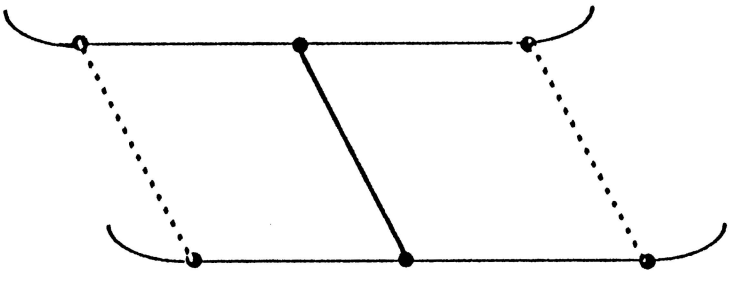


Il nous faut donc retrouver l'arête de tangence à partir d'une seule des deux surfaces, puis éventuellement relancer l'exploration à partir de cette arête pour retrouver d'autres surfaces, qui pourraient d'ailleurs induire d'autres arêtes de tangence. On doit relancer les recherches ainsi jusqu'à ce qu'aucun nouvel objet (arête ou surface) ne soit créé. En pratique il est rare de faire plus de deux passages.

Comment déterminer les arêtes de tangence susceptibles de se trouver sur une surface ? Pour cela nous allons intercaler quelques instructions supplémentaires dans la phase de l'algorithme qui cherche toutes les arêtes d'une surface.

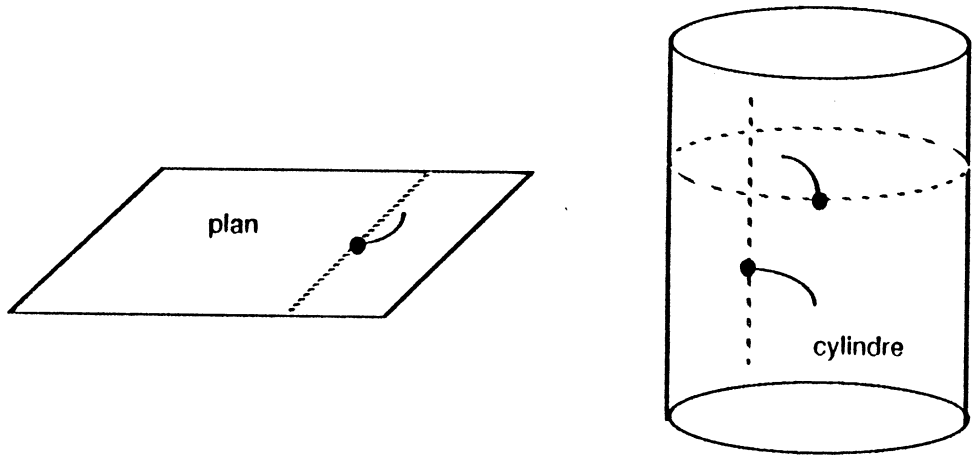
Quand nous examinons toutes les arêtes qui entourent un sommet pour ne retenir que celles qui appartiennent à la surface nous allons aussi chercher à identifier toutes les arêtes qui partent tangentiellement à la surface. Pour cela il suffit de faire le produit scalaire entre la normale à la surface au sommet et la tangente à l'arête, s'il est nul la tangente est dans le plan tangent. Nous mettons de coté le sommet et la direction de la tangente pour un traitement ultérieur.

figure IV-12 arêtes tangentes à la surface



Une fois la recherche des arêtes de la surface terminée nous nous trouvons avec une liste de sommets sur la surface, à chacun de ces sommets est associée une direction de tangente; remarquons qu'un même sommet peut se trouver deux fois dans cette liste avec deux directions de tangente différentes. Il s'agit maintenant de trouver des arêtes de tangence qui relient ces sommets. Cette recherche est très spéculative et repose sur les cas les plus courant d'arêtes de tangence. En fait nous examinons chacun des sommets pour lui associer une courbe sur la surface qui passe par le sommet en étant perpendiculaire à la direction de tangente et qui est donc susceptible de porter l'arête de tangence. Puis nous regroupons les courbes identiques pour obtenir sur chaque courbe un ensemble de sommets qui vont découper la courbe en arête. Les courbes sont cataloguées suivant le type de surface et la direction de la tangente, il s'agit encore de procédures dédiées prises en charge par le modeleur surfacique; nous cherchons en fait des courbes de types géodésiques. Ainsi pour le plan ce sont des droites, pour les surfaces de révolution des méridiens ou des parallèles.

figure IV-13 courbes portants les arêtes de tangence

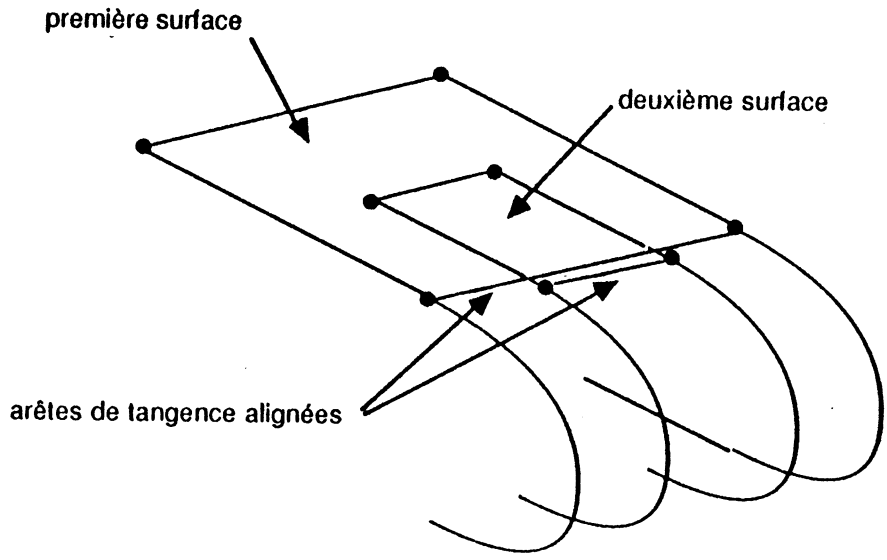


Pour critiquer cette méthode nous dirons qu'elle est loin d'être générale puisqu'elle se rapproche nettement de méthodes de reconnaissance de formes par stéréotypes. Plus précisément nous nous appuyons sur certaines caractéristiques locales pour choisir une forme globale parmi un petit ensemble de stéréotypes connus et faciles.

De plus nous pouvons remarquer que l'algorithme n'est pas très efficace car il va rechercher souvent des arêtes où il s'en trouve déjà. Par exemple, chaque arête de tangence peut être découverte à partir des deux surfaces qui la contiennent. Ce n'est que parce qu'il y a en général un faible nombre d'arêtes de tangence par rapport au nombre total d'arêtes que cette recherche ne grève pas. les performances globales.

Dernière remarque : il peut arriver que nous introduisions des arêtes alignées comme nous les avions déjà décrites. Dans le cas de la figure cela vient du problème déjà évoqué de connexité des ensembles d'arêtes sur les surfaces. Nous allons laisser ces arêtes en place sans chercher à les détecter (c'est très couteux), mais par la suite nous montrerons comment il est possible lors de la recherche des faces de se débarrasser de ces arêtes indésirables.

figure IV-14 arêtes de tangence alignées

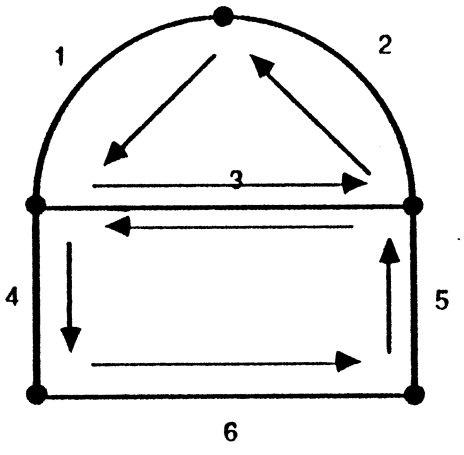


IV-3-3 Recherche des faces

Nous avons maintenant identifié toutes les surfaces de notre fil-de-fer et nous connaissons toutes les arêtes. Il nous faut chercher les faces. Nous connaissons avec chaque surface une liste d'arêtes qui forment un graphe dessiné sur la surface; remarquons que ce graphe n'est pas forcément planaire mais qu'il est de genre inférieur ou égal à celui de la surface, ainsi sur le tore le graphe peut être de genre 1. (Rappelons que le genre d'un graphe est le plus petit genre d'une surface sur laquelle on peut dessiner le graphe sans intersection d'arêtes).

La première étape de notre recherche de faces est de trouver les circuits formés par les arêtes qui sont les éléments des frontières des faces. Pour cela il faut parcourir notre graphe en choisissant à chaque sommet de tourner dans le sens anti-trigonométrique par rapport à la normale. Ce choix nous garantit que nous passons d'une arête à une autre arête qui limite la même face.

figure IV-15 recherche de circuits



successeurs	
-6	-4
-5	-6
-4	-1
-3	4
-2	-5
-1	-2
1	3
2	1
3	2
4	6
5	-3
6	5

tri des arêtes

Nous commençons donc par trier à chaque sommet les arêtes dans le sens anti-trigonométrique autour de la normale. Le résultat de ce tri est d'attribuer à chaque arête un successeur, remarquons que nous donnons un signe aux identificateurs d'arête pour distinguer les deux sens de parcours de l'arête. Soit n le vecteur normal à la surface, a1 la première arête arbitrairement choisie comme origine du tri, ai et aj deux arêtes que nous voulons comparer. Nous voulons savoir si le sens trigonométrique est $\mathbf{a_1, a_i, a_j}$ ou $\mathbf{a_1, a_j, a_i}$. Nous utilisons les tangentes aux trois arêtes $\mathbf{t_1, t_i, t_j}$.

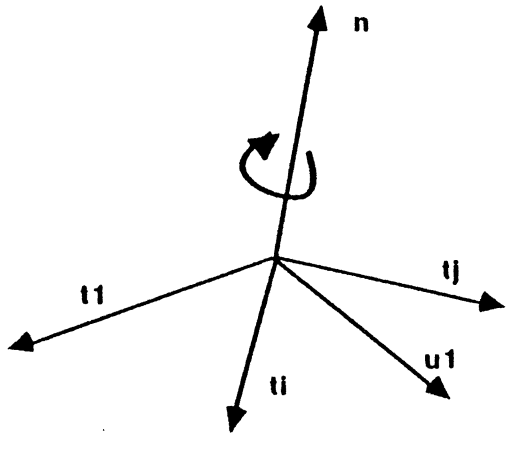
Avant toute comparaison nous calculons $\mathbf{u_1 = n \times t_1}$, ainsi $\mathbf{t_1, u_1, n}$ forment un repère orthonormé. Pour comparer $\mathbf{a_i}$ et $\mathbf{a_j}$ nous calculons :

$s_i = t_i . u_1$ et $s_j = t_j . u_1$
si $s_i \geq 0$ et $s_j \leq 0$ alors a_i est avant a_j
si $s_j \geq 0$ et $s_i \leq 0$ alors a_j est avant a_i
si s_i et s_j sont de même signe nous calculons
$c_i = t_i . t_1$ et $c_j = t_j . t_1$
si s_i et $s_j \geq 0$ alors
si $c_i \geq c_j$ alors a_i est avant a_j
si $c_j \geq c_i$ alors a_j est avant a_i
si s_i et $s_j < 0$ alors
si $c_i \leq c_j$ alors a_i est avant a_j
si $c_j \leq c_i$ alors a_j est avant a_i
sauf dans ce dernier cas si $c_i = 0$ alors a_i est avant a_j

-Remarquons que quand nous écrivons $x \leq 0$, $x > 0$, ou $x = 0$ nous entendons en fait

$x < e$, $x > e$, $-e < x < e$. où e est un réel petit comme nous l'avons déjà expliqué.

figure IV-16 tri des arêtes



- Notons encore que les deux tangentes t_i et t_j peuvent être colinéaires; dans ce cas s'il s'agit d'arcs de cercles il faut comparer les rayons (c'est à dire les courbures) pour les départager. S'il s'agit de deux segments ou de deux arcs de cercles de même rayon nous nous trouvons en présence d'arêtes alignées comme nous les avions signalées. Dans ce cas nous

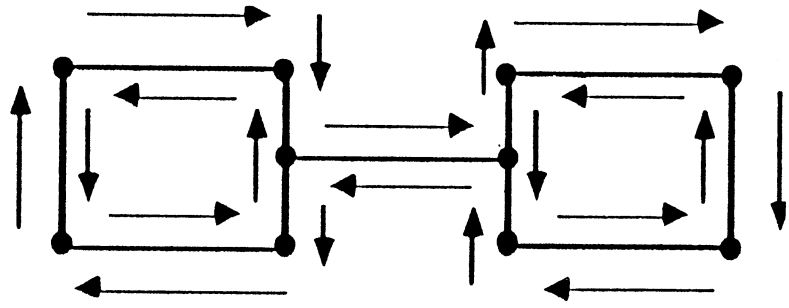
allons supprimer une des arêtes, c'est à dire que nous allons modifier notre ordre de succession des arêtes pour qu'elle ne soit pas prise en compte dans la recherche des faces.

Quelle arête supprimer entre ai et aj : nous comparons leurs longueurs et nous supprimons la plus longue. Si les arêtes sont de même longueur il est indifférent de supprimer l'une ou l'autre, cependant il faut que la même soit supprimée quand nous trions autour de l'autre extremité, pour cela nous supprimons arbitrairement l'arête de plus grand identificateur.

recherche des circuits

Chaque arête orientée possède maintenant un successeur qui est une autre arête orientée. Nous allons parcourir le graphe de successeur en successeur pour construire les circuits. La recherche s'arrête quand chaque arête appartient à deux circuits, un dans chaque sens. Si les deux circuits qui passent par une arête sont identiques, l'arête est intérieure à une face et doit être supprimée. De telles arêtes sont appelées isthmes, car si on supprime l'arête on déconnecte le graphe.

figure IV-17 un isthme



construction des faces

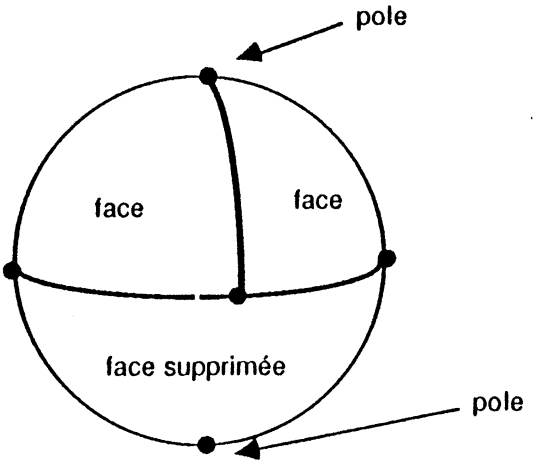
Les circuits ne suffisent pas, il nous faut encore les assembler pour découvrir les faces. En effet une face peut être limitée par plusieurs circuits. Chaque circuit définit une région sur la surface et ce sont les intersections de ces différentes régions qui forment les faces, en supprimant les régions infinies.

- Sur les surfaces infinies (plan, cylindre, cône ...) il est facile de déterminer pour chaque circuit s'il délimite une région finie ou infinie. Il suffit de chercher une arête du circuit la plus éloignée dans une direction puis de déterminer à partir de l'orientation de cette arête si l'infini se trouve à l'intérieur ou à l'extérieur de la région. Une fois les circuits départagés en finis et infinis nous créons une face pour chaque circuit fini, puis nous cherchons si les circuits infinis sont contenus dans ces faces; il suffit de choisir un sommet du circuit et de le classifier par rapport à la face. Quand un circuit infini se trouve à l'intérieur d'une face il correspond à un trou dans la face.

- Sur les surfaces finies (sphère, tore) chaque circuit délimite une région finie et est donc candidat à former une face. Pour démêler la complexité de ce cas nous nous reposons sur la présence des arêtes silhouettes qui doivent découper notre surface.

En particulier nous appelons pôles de la surface les points extrêmes de la surface dans les directions de projections; par un pôle doivent obligatoirement passer des arêtes silhouettes si le pôle appartient à une face. Ceci nous donne un critère pour éliminer tous les circuits qui contiennent un pôle intérieur. Nous utilisons encore une autre heuristique pour le tore qui est d'éliminer les circuits qui font le tour complet de la surface, car les vraies faces doivent emprunter des arêtes silhouettes qui limitent à des demi-tours.

figure IV-18 suppression des faces contenant un pôle

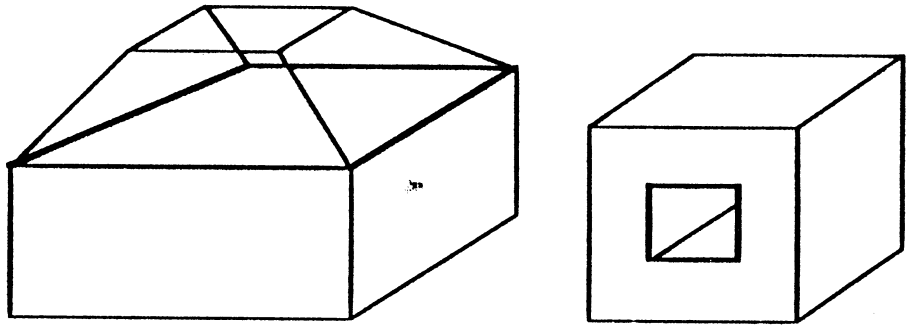


IV-3-4 Intersection de faces

Ici encore nous allons créer des éléments supplémentaires. Il s'agit ici de pseudo-faces qui peuvent avoir une double origine :

- Soit elles contiennent des pseudo-arêtes, soit elles sont formées de vraies arêtes mais résultent d'alignement fortuits. De telles faces ont la particularité de se trouver entièrement dans le vide ou dans la matière. Ces pseudo-faces peuvent avoir entre elles ou avec les vraies faces des intersections illicites.

figure IV-19 pseudo-faces



Nous ne savons pas encore distinguer les vraies faces des pseudo-faces, mais nous pouvons rechercher les intersections. Markowsky et Wesley distinguent deux types d'intersections :

- Type I une arête perce une face en un point intérieur à la face. Alors, soit l'arête est une pseudo-arête soit la face est une pseudo-face (soit les deux).

- Type II deux faces s'intersectent suivant une courbe intérieure aux faces qui n'est pas une arête. Alors l'une au moins des deux faces est une pseudo-face.

Nous allons recenser ces intersections et les conserver sous formes de contraintes logiques qui doivent être vérifiées par les solides que nous reconstruisons.

Pour détecter les intersections de type I il faut intersecter les arêtes avec les surfaces ce qui donne des points, puis chercher si ces points sont intérieurs à des faces.

Pour détecter les intersections de type II il faut intersecter les surfaces, ce qui donne des courbes, puis chercher si ces courbes sont intérieures à des faces.

Remarquons que la recherche de ces intersections est coûteuse en temps, nous avons preferré la rendre optionnelle, car nous avons remarqué qu'elle était souvent inutile. C'est à dire que les algorithmes fonctionnent en général sans considérer ces intersections qui ne sont déterminantes que dans des cas particuliers très symétriques assez éloignés de problèmes réels.

IV-3-5 Construction des solides

Algorithme de construction

Nous abordons la dernière étape de notre algorithme qui est de déterminer tous les solides constructibles à partir de l'ensemble de faces

déterminées aux étapes précédentes et compatible avec les projections ou le fil-de-fer donnés en entrée.

Voyons tout d'abord les différents états que nous pouvons affecter à une face :

- Une face reçoit un état parmi les trois possibles -1, 0, 1 nous ajoutons l'état -2 qui signifie que l'état de la face n'est pas encore déterminé.

- 0 signifie que la face est une pseudo-face et ne fait pas partie de la frontière du solide en cours de construction.

- 1 signifie que la face appartient à la frontière du solide en conservant son orientation actuelle, toutes les faces ayant été construites en conservant l'orientation naturelle de la surface qui les contient.

- -1 signifie que la face appartient à la frontière du solide mais en inversant son orientation.

Chaque arête du pseudo-fil-de-fer appartient à un certain nombre de faces, si toutes ces faces sont déterminées l'arête peut avoir trois états :

- Pseudo-arête si toutes les faces incidentes à l'arête sont des pseudo-faces.

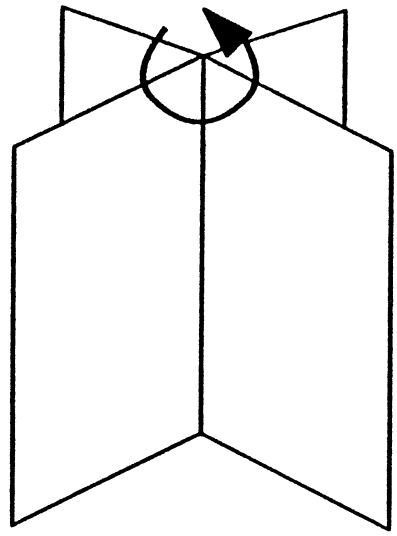
- arête-fantôme si il n'y a que deux faces réelles qui contiennent l'arête mais que ces deux faces sont sur la même surface.

- arête réelle dans tous les autres cas.

Cependant les faces qui entourent l'arête doivent vérifier les conditions de Moëbius : il doit y en avoir un nombre pair de réelles et leurs orientations doivent être compatibles. Pour cela, s'il y a plus de trois faces autour de l'arête nous devons les trier autour de l'arête dans un sens. Ce tri correspond à l'ordre dans lequel un petit cercle autour de l'arête rencontrerait les faces. Pour faire ce tri nous procédons exactement comme pour le tri des arête autour des sommets que nous avons fait au

dessus. Cette fois le vecteur de référence est un vecteur tangent à l'arête, et nous trions les vecteurs normaux aux faces.

figure IV-20 tri des faces autour d'une arête



Si notre solide vérifie les conditions de Moëbius il doit être aussi compatible avec les données initiales. Il y a en effet un très grand nombre de solutions compatibles avec les seules conditions de Moëbius. Les conditions à vérifier sont les suivantes :

- Si nous sommes partis d'un modèle fil-de-fer il faut que chaque arête de ce fil-de-fer soit une arête réelle ou une arête fantôme, mais aucune ne doit être une pseudo-arête.

- Si nous sommes partis de projections, il faut que parmi les arêtes qui se projettent sur une ligne il y en ait au moins une de réelle pour justifier la présence de la ligne.

Nous allons maintenant décrire un algorithme qui explore les différents solides possibles pour énumérer ceux qui sont compatibles avec les contraintes. Cet algorithme repose sur le schéma suivant :

- Choisir une face et l'orienter arbitrairement.

- Propager l'orientation de cette face par les règles de Moëbius en vérifiant les contraintes.

- s'il n'y a plus de propagation possible retourner à la première étape.

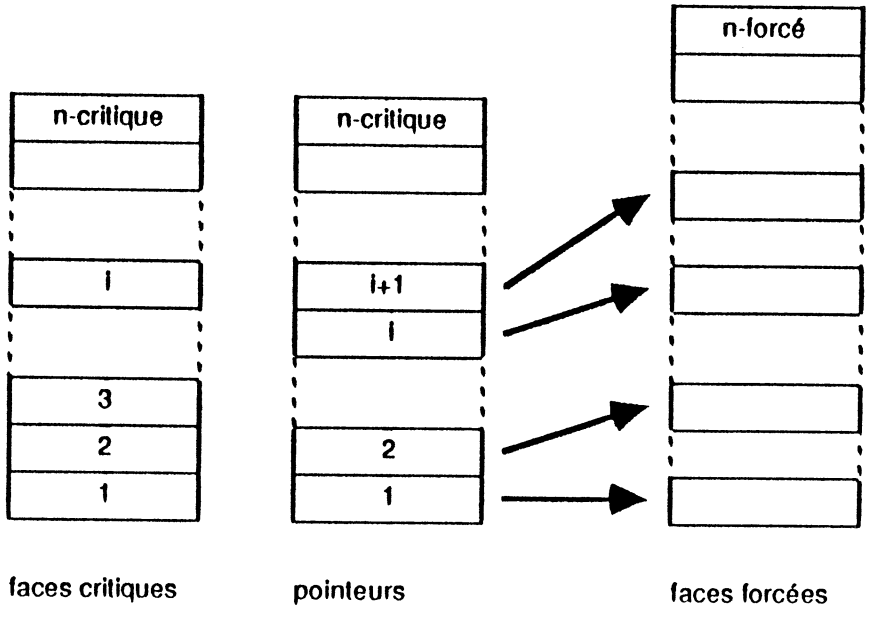
- si une contrainte est violée revenir en arrière sur une face et changer son orientation.

Ceci jusqu'à ce que toutes les faces soient orientées.

Avant d'entrer dans les détails de l'algorithme nous allons décrire quelques éléments de structure de données.

- Appelons faces critiques les faces orientées arbitrairement sur l'orientation desquelles nous pouvons revenir. Nous enregistrons ces faces dans une table des faces critiques, critiques(n-critique) où n-critique est le nombre de faces critiques.

- A chaque face critique est associée un ensemble de faces forcées dont l'orientation est déterminée par celle de la face critique. Les faces forcées sont rangées dans un tableau forcées(n-forcée), les faces forcées par la face critique nc sont les faces du tableau forcées comprises entre les indices forcé-critique(nc) et forcé-critique(nc+1)-1.



Dans le cas des projections nous associons aussi à chaque ligne un nombre entier n-arête(ligne) qui est le nombre d'arêtes réelles ou indéterminées se projetant sur la ligne. A chaque arête disparue (pseudo-arête ou arête fantôme) nous décrémentons ce nombre pour toutes les lignes sur lesquelles se projette l'arête. Dès que ce nombre devient nul pour une ligne cela signifie qu'une contrainte est violée et qu'il faut revenir en arrière.

L'algorithme est présenté ci-dessous en notation algorithmique (avec des instructions aller-à et des labels).

```
{ initialisation }

pour F:= 1 à n-face faire état(F) := -2; { toutes les faces sont indéterminées}
n-critique := 0;
n-forcé    := 0;

{ recherche d'une face de départ indéterminée, et orientation}

recherche:
chercher-face( F );
si toutes les faces sont déterminées alors
    montrer-résultat;          { on a trouvé un solide solution }
    aller-à retour-arrière     { on cherche les solutions suivantes }
fin

{ enregistrement de la face critique et initialisation du forçage }

init-forçage:
n-critique := n-critique + 1;
critique(n-critique) := F;
n-forcé := n-forcé + 1;
forcés(n-forcé) := F;
forcé-critique(n-critique) := n-forcé;
N := n-forcé;


{ forçage à partir de la face suivante forcés(N) }

forçage:
F := forçés(N);
pour A := toutes les arêtes de F faire
    examiner la liste des faces autour de A;
    si il y a une seule face indéterminée G alors
            calculer état(G);          { avec la règle de Moëbius }
            n-forcé := n-forcé + 1;    { G est une face forcée par F }
            forcés(n-forcé) := G;
    fin
```

```
    si il n'y a aucune face indéterminée alors
            calculer l'état de A;
            vérifier les contraintes pour A;
            si une contrainte est violée alors
                    aller-à retour-arrière
            fin
    fin
fin
N := N + 1;
si N ≤ n-forcé alors aller-à forçage;

{ s'il n'y a plus de faces  forcées on cherche une autre face }

aller-à recherche;

{ retour arrière sur la dernière face critique }

retour-arrière:
F := critiques(n-critiques);

{ on remet à indéterminé toutes les faces forcées par F }
pour N:= forcé-critique(n-critique) + 1 à forcé-critique(n-forcé) - 1
    état (forcés(N)) := -2;
fin
n-forcé := forcé-critique(n-critique) - 1;


{ si F était orientée on recommence avec F pseudo-face }

si état (F) <> 0 alors
    état(F) := 0;
    aller-à init-forçage;
fin;

{ si c'était une pseudo-face on continue à revenir en arrière }

état(F) := -2;
n-critique := n-critique - 1;
si n-critique > 0 alors aller-à retour-arrière ;

{ si il n'y a plus de faces critiques, l'exploration est terminée }
```

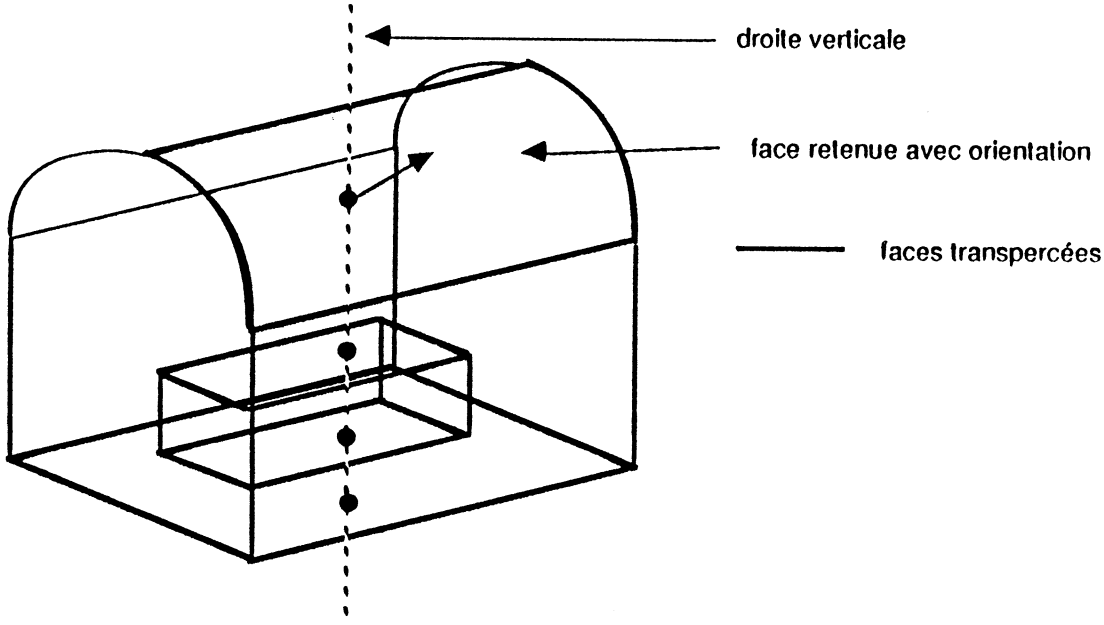
Nous allons maintenant expliciter quelques une des opérations mises en oeuvre dans cet algorithme.

Recherche et orientation d'une face critique.

Il s'agit de trouver une face indéterminée et de lui donner une orientation pour pouvoir continuer la recherche. Quand nous choisissons une face indéterminée et que nous supposons que ce n'est pas une pseudo-face, nous pouvons l'orienter telle qu'elle serait dans le solide. Pour cela nous choisissons une face extrême dans une direction.

Plus précisément considérons une droite verticale (parallèle à l'axe Z par exemple), cette droite perce un certain nombre de faces que nous pouvons trier par rapport à l'altitude du point d'intersection. Pour cela il faut demander au modeleur géométrique de calculer l'intersection de la droite avec chaque surface puis chercher sur chaque surface la face qui contient le point d'intersection (s'il y en a une). Nous choisissons la face indéterminée de plus grande altitude. Nous sommes capables pour cette face de déterminer si la matière se trouve au-dessus ou au-dessous de la face. En effet, les points les plus hauts de la droite sont naturellement dans le vide, puis, pour chaque face située au-dessus de celle qui nous intéresse, soit il s'agit d'une pseudo-face et l'état vide-matière ne change pas, soit il s'agit d'une face réelle et on change d'état vide ou matière. Nous pouvons ainsi redescendre à l'état de la face indéterminée puisqu'il n'y a que des faces déterminées au-dessus. Pour orienter la face nous faisons le produit scalaire de la direction de la droite avec la normale à la face, et nous orientons suivant la position de la matière.

figure IV-21 sélection de la première face critique



Forçage :

L'opération de forçage consiste à propager l'état des faces de proche en proche. Plus précisément quand toutes les faces incidentes à une arête sont déterminées sauf une, nous pouvons déduire l'état de cette dernière. On considère les deux vraies faces (non pseudo-faces) qui entourent la face indéterminée dans l'ordre de rotation autour de l'arête. Si ces deux faces tournent dans le même sens par rapport à l'arête il faut orienter la face dans le sens opposé pour respecter la règle de Moëbius. Si les deux faces tournent dans des sens opposés il faut faire une pseudo-face de la face indéterminée. Dans chacun de ces cas on vérifie bien la règle : deux vraies faces consécutives ont des orientations opposées.

Vérification des contraintes

Pour le problème des fil-de-fer la seule contrainte à vérifier est qu'aucune arête ne disparaisse parce que toutes les faces qui lui sont incidentes sont des pseudo-faces.

Pour les projections il y a deux manières pour une arête de disparaître : ce peut être une pseudo-arête (toutes les faces sont des pseudo-faces), ou une arête fantôme (l'arête est partagée par deux faces cosurfaciques). Dans ces deux cas l'arête ne peut plus se projeter sur une ligne.

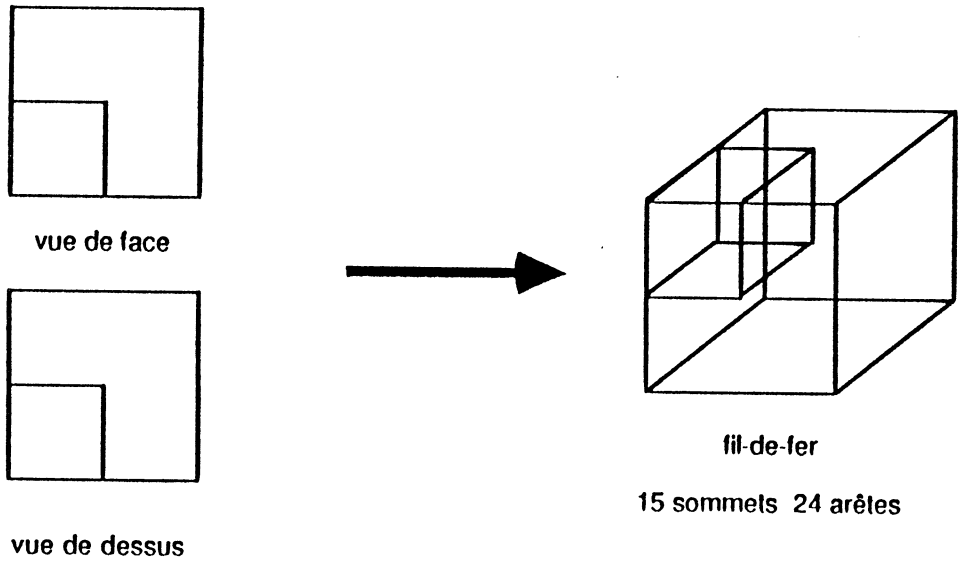
Nous dirons qu'une contrainte est violée quand toutes les arêtes se projetant sur une ligne disparaissent ainsi, alors la présence de la ligne sur les projections n'est plus compatible avec la solution.

Pratiquement nous associons avec chaque ligne un entier qui est le nombre d'arêtes se projetant sur la ligne. A chaque fois que dans l'étape de forçage nous déterminons qu'une arête est une pseudo-arête ou une arête fantôme nous décrémentons le compte de chacune des lignes sur lesquelles se projette l'arête. Si ce compte devient nul pour l'une des lignes une contrainte est violée.

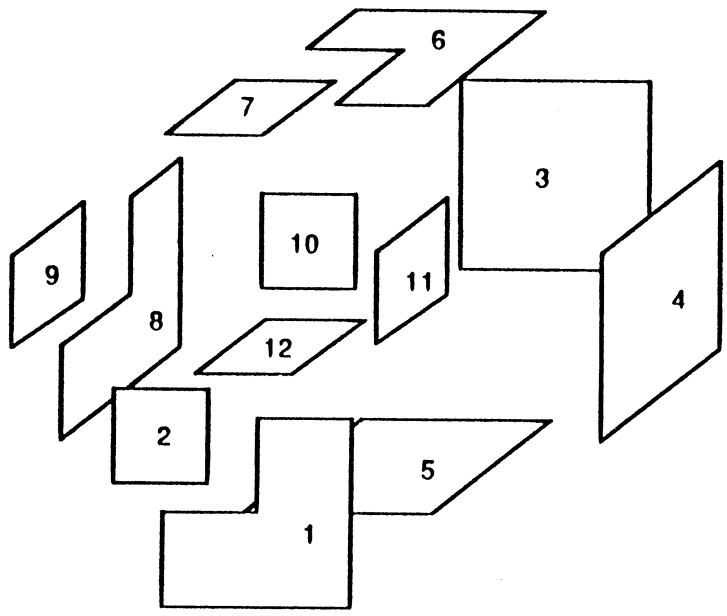
Sur la figure suivante nous montrons les différentes étapes d'une reconstruction d'un solide.

figure IV-22 exemple de reconstruction

sommets et arêtes



recherche des faces



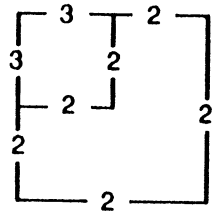
Première face critique : 6 orientée vers le haut.

Faces forcées : 3, 4, 1, 8 directement, puis 5 (toutes orientées vers l'extérieur)

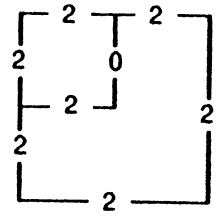
Deuxième face critique : 7 orientée vers le haut.

Faces forcées : 2, 9 orientées 10, 11 pseudo-faces, ici une contrainte est violée l'arête entre les faces 1, 2 et 11 est une arête fantôme, celle entre 10 et 11 est une pseudo-arête. On peut donc arrêter le forçage avant d'aborder la face 12.

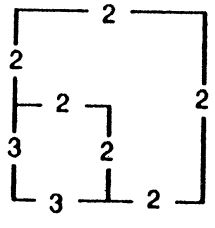
comptes d'arêtes par ligne au début et avant de forcer la face 12



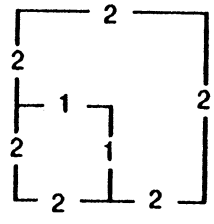
vue de face



vue de face



vue de dessus



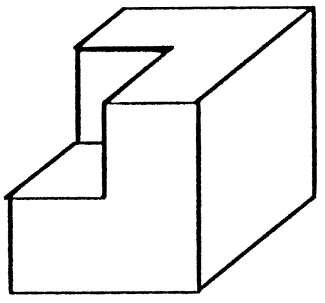
vue de dessus

Nous devons revenir en arrière sur la dernière face critique 7 et remettre indéterminée les faces 2, 9, 10 et 11.

Face critique : 7 (pseudo-face)

Faces forcées : 2, 9 (pseudo-faces) 10, 11, 12 (orientées)

Il n'y a pas de contrainte violée et toutes les faces sont déterminées, la première solution est le solide limité par les faces : 1, 3, 4, 5, 6, 8, 10, 11, 12

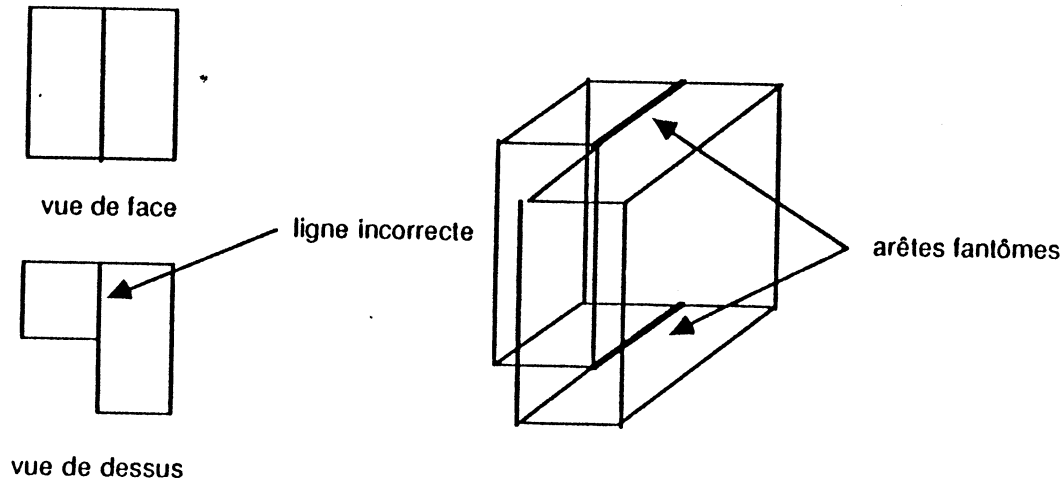


On peut continuer la recherche et revenir sur la face 6, en faire une pseudo-face, alors 3, 4, 1, 9 deviennent des pseudo-faces et il y a de nombreuses arêtes disparues.

Relâchement de la contrainte sur les lignes.

Il peut arriver que nos projections contiennent des lignes supplémentaires introduites par erreur, alors l'algorithme ne pourra pas découvrir de solides qui rende compte de la présence de ces lignes. Cependant quand une contrainte de ligne est violée il s'agit en fait de tout un groupe de lignes qui disparait, en effet quand l'algorithme de forçage prend une mauvaise direction il supprime ou ajoute tout un bloc de matière (le petit cube ou le grand cube sur l'exemple précédent). Si nous ne trouvons pas de solide nous pouvons relancer une recherche en autorisant la disparition d'une seule ligne; nous ne pourrons pas laisser disparaître un bloc qui entraînerait plusieurs lignes; par contre nous pourrons faire disparaître une ligne incorrecte. Nous pouvons continuer à augmenter ce nombre de lignes disparues en sachant qu'au delà d'un certain seuil (qui dépend du cas) ce seront des vraies lignes qui disparaîtrons.

figure IV-23 projections avec une ligne incorrecte



La solution avec les deux arêtes fantômes entraîne la disparition de la ligne incorrecte.

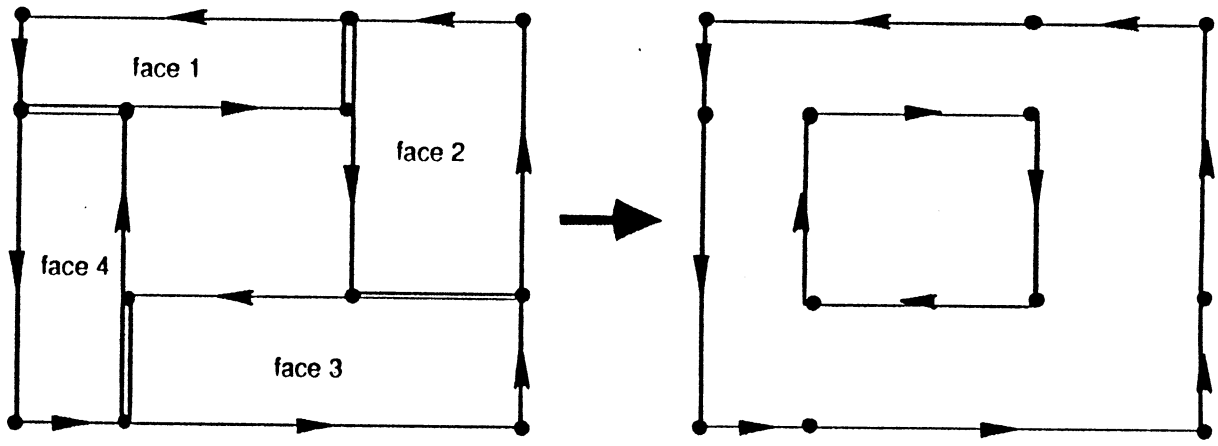
Nous sommes maintenant capables d'énumérer tous les solides correspondant à un jeu de projections ou à un fil-de-fer. Pour exploiter ces solides il nous reste quelques opérations à accomplir.

Derniers traitements

Fusion des faces cosurfaciques

Il s'agit en fait de supprimer les arêtes fantômes : une arête fantôme sépare deux faces situées sur la même surface, nous pouvons regrouper ces faces en une seule. Pour cela nous procédons en deux étapes . D'abord nous regroupons ensemble toutes les faces d'une même surface qui ont des arêtes communes. Ensuite nous supprimons dans le graphe ainsi obtenu les arêtes en double qui sont parcourues dans les deux sens et nous reconstruisons une seule face en cherchant les circuits du graphe.

fusion de faces



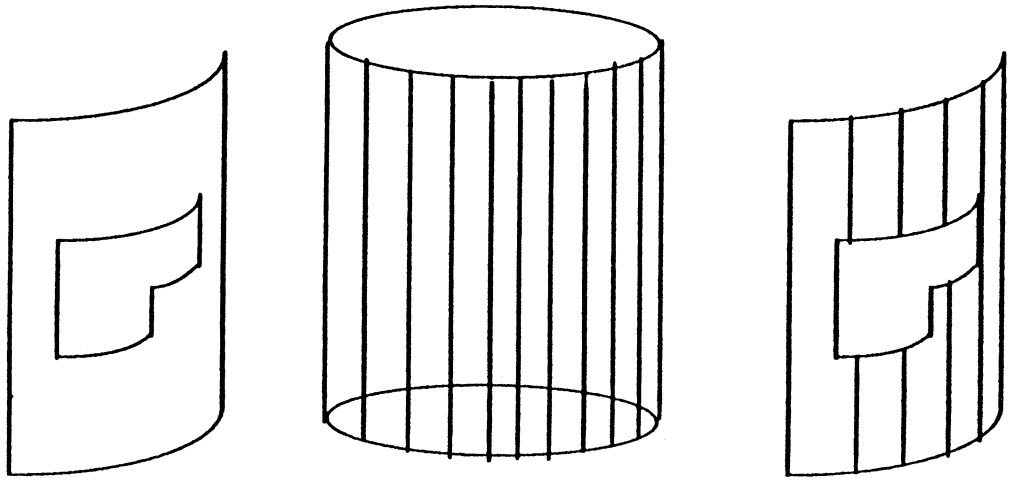
Polyédrisation

Pour pouvoir utiliser les solides que nous avons crées dans un système de C.A.O comme EUCLID il nous faut encore en donner une approximation polyédrale (voire dessins en annexe). Cette polyédrisation se fait en deux étapes.

- Polyédrisation de chaque face, cette polyédrisation repose sur une facettisation de la surface qui porte la face. Celle-ci est découpée en

bandes si c'est un cylindre, en triangles si c'est un cône, en carrés si c'est une sphère ou un tore, puis tous les éléments à l'intérieur de la face sont gardés et ceux qui sont sur la frontière sont découpes (figure IV-25).

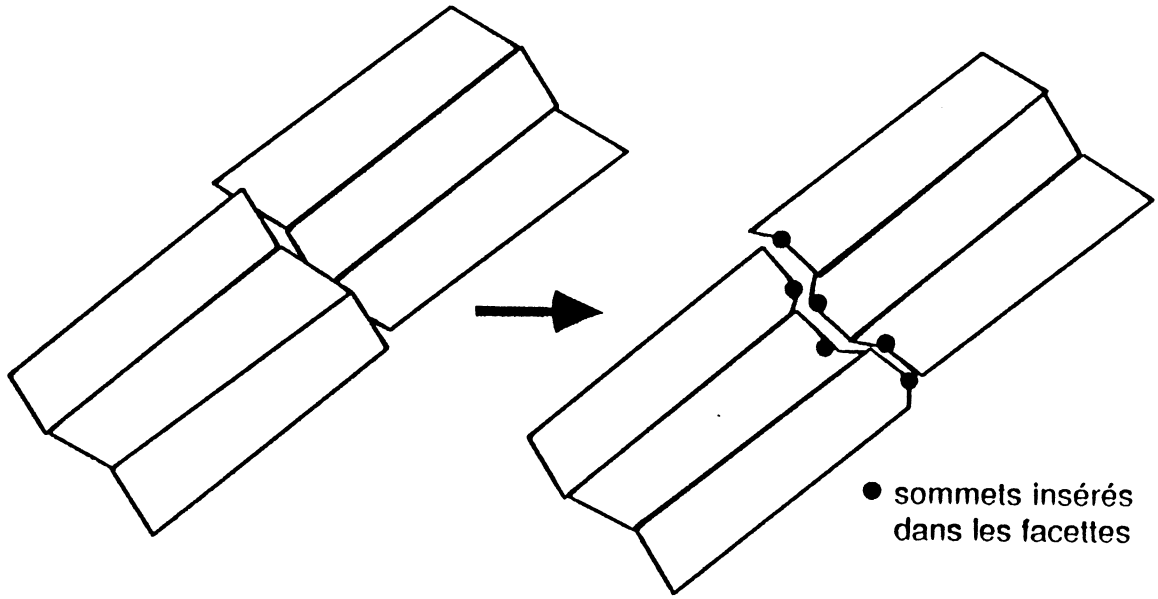
figure IV-25 polyédrisation d'une face



une face cylindrique | polyédrisation du cylindre | face polyédrisée

- Raccordement entre les faces, le long des arêtes du solide les facettes de la polyédrisation ne se raccordent pas exactement. Pour avoir des polyèdres parfaitement corrects il faut encore insérer dans chaque facette qui s'appuie sur l'arête les points des facettes en vis-à-vis ce qui les rend légèrement gauche (figure IV-26). On pourra constater sur les figures en annexes que cette opération n'a pas été réalisée.

figure IV-26 raccord des facettes



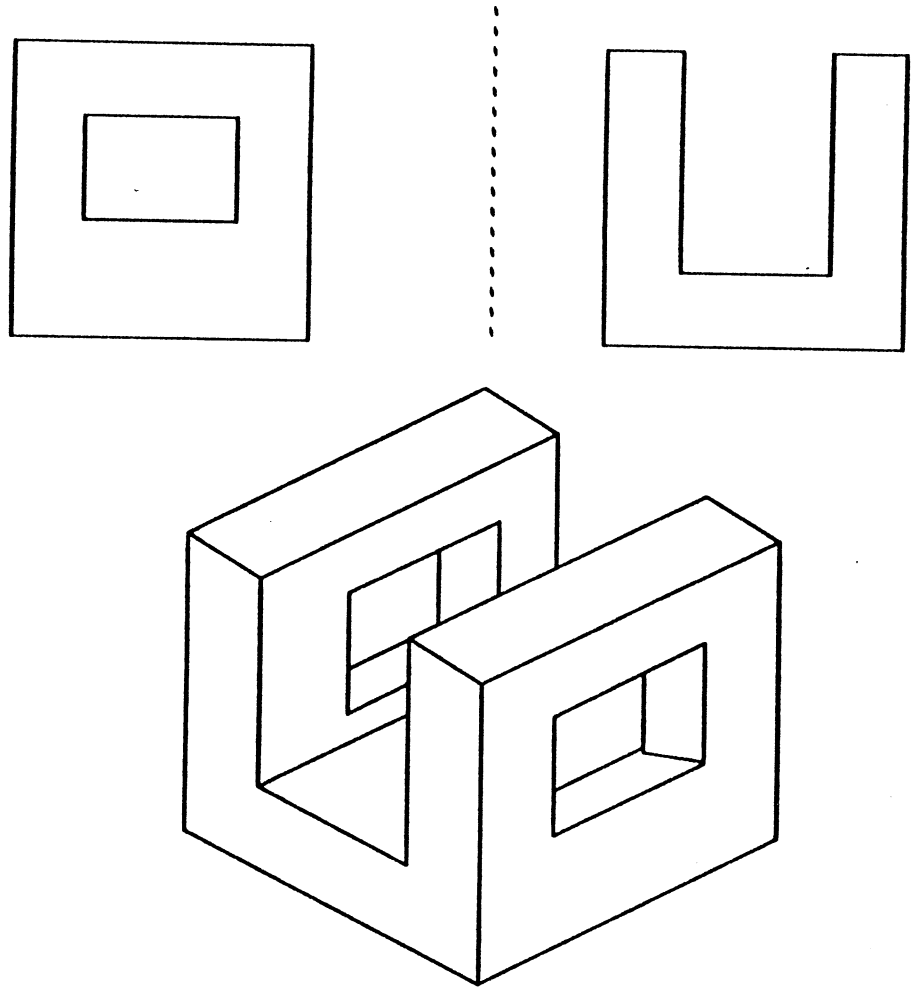
IV-4 Intersection de prismes

Nous avons proposé dans la première partie d'utiliser ces algorithmes comme outils interactifs de création de solides. Nous allons donner maintenant un exemple d'application allant dans ce sens. Il s'agit de construire un solide défini par l'intersection de deux prismes droits infinis orthogonaux.

IV-4-1 Présentation du problème

Nous nous donnons deux faces planes chacune située dans un plan de réferrence (x=0, y=0, z=0), chacune des faces est constituée de un ou plusieurs contours (face trouée) , nous considérons alors le solide formé par l'intersection des deux prismes infinis obtenus à partir des faces en balayant le long de la direction perpendiculaire à la face. (figure IV-27).

figure IV-27 Intersection de deux prismes



Cette construction est utilisée en C.A.O pour définir des solides et on utilise en général dans ce cas les opérateurs de balayage et d'intersection. Le balayage est naturellement fini, ce qui implique de faire attention à donner la bonne extension au prisme et d'ajouter quatre surfaces inutiles (les extrémités des prismes). D'autre part l'algorithme d'intersection est trop général pour ce cas particulier qui pose des problèmes car les surfaces des deux prismes sont souvent tangentes ou confondues. On peut ici remplacer un application hasardeuse et coûteuse d'un algorithme général par un algorithme plus performant.

La marche suivie par notre algorithme est de se ramener au cas où l'on connait deux projections du solide pour ensuite appliquer ce qui a déjà

été fait. Il nous suffit donc de trouver un algorithme qui calcule les projections orthogonales de l'intersection des deux prismes.

IV-4-2 Projections de l'intersection de deux prismes

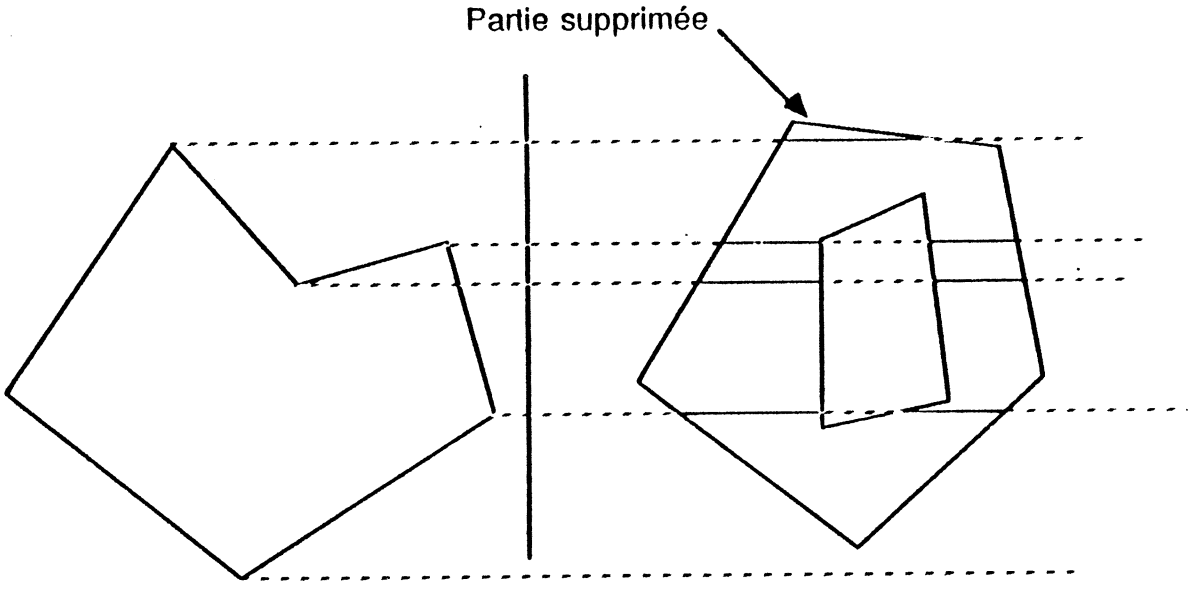
Les projections orthogonales sont celles du fil-de-fer, il faut donc que nous déterminions les projections des arêtes de l'intersection. Nous appelerons bases les deux faces initiales qui définissent les prismes. Examinons ce que peuvent être les arêtes de l'intersection de deux solides, il y en a deux types :

- Arêtes provenant des solides initiaux, éventuellement tronquées.

- Arêtes provenant de l'intersection des faces des solides initiaux.

Voyons ce que sont ces deux familles d'arêtes dans notre cas. Les arêtes des deux prismes sont des droites infinies perpendiculaires aux bases issues des sommets des bases, les projections de ces arêtes sont des droites parallèles horizontales ou verticales. Dans l'intersection des deux prismes ces droites sont limitées par la base. (figure IV-28).

figure IV-28 projection de l'intersection



Les arêtes provenant de l'intersection des faces des deux prismes se projettent sur les arêtes des bases. Mais il faut retirer sur chaque base les portions de contour qui se trouvent à l'extérieur du prisme définit par l'autre base.

En résumé pour construire les projections de l'intersection de deux prismes on trace sur chaque base un réseau de droites parallèles issues des sommets de l'autre base, on conserve les parties de ces droites qui sont à l'intérieur de la base et on retire les parties de la base qui sont à l'extérieur de ce réseau.

Il ne reste plus qu'à faire tourner notre algorithme sur ces projections, on remarquera que les deux vues suffisent à reconstituer le solide. Un exemple d'application de cet algorithme se trouve dans les exemples en annexe.

V Conclusion

Cette conclusion contient deux parties distinctes, d'abord une revue des travaux académiques déja effectués sur le sujet comparés à celui-ci. Puis la conclusion proprement dite qui établira les limites et les possibilités d'extension de ce travail.

V-1 Revue des travaux publiés sur le sujet

Tous les travaux que nous allons citer maintenant sont ceux qui sont parvenus à notre connaissance par les articles qui sont dans la bibliographie. Il se peut, bien sur, que nous en ayons oublié.

Nous avons divisé ces articles en quatre groupes :

- Les précurseurs sont les plus anciens mais la plupart sont incomplets.

L'approche topologique correspond à ce que nous avons décrit dans la deuxième partie de ce rapport.

L'approche reconnaissance de forme est à rapprocher de la géométrie constructive puisqu'elle essaie de trouver une construction du solide à partir des plans.

L'approche frontière est celle que nous avons suivie.

V-1-1 Les précurseurs

Les premiers travaux sur le sujet sont ceux d'Idesawa, Lafue et Preiss **[Idesawa 73,75], [Lafue 76,78], [Preiss 80,81,84]** celui de Preiss étant le plus récent et le plus complet. Ces travaux sont proches de l'approche frontière, c'est-à-dire que les auteurs cherchent à reconstituer sommets, arêtes et faces du solide. Cependant ils ne distinguent pas l'étape du fil-de-fer intermédiaire à partir de laquelle redémarre la recherche des faces. Idesawa et Lafue cherchent à éliminer les sommets, arêtes et faces supplémentaires en appliquant des règles du style chaque sommet appartient au moins à trois arêtes. Ces techniques heuristiques

sont cependant coûteuses et incomplètes. Preiss essaie de reconnaître directement les faces sur les projections mais sa méthode semble extrêmement coûteuse d'après les exemples qu'il donne. Tous ces auteurs n'abordent le problème que pour les solides polyédriques.

V-1-2 L'approche topologique

Cette approche que nous avons décrite dans la deuxième partie consiste à se limiter d'abord au problème du fil-de-fer et à appliquer des algorithmes de reconnaissance de graphes planaires. On peut se reporter à la deuxième partie et à **[Ganter & Ulcker], [Courter & Brewer], [Hanrahan].**

V-1-3 L'approche reconnaissance de forme

L'idée de cette approche est de reconnaitre sur les projections des primitives et de combiner ces primitives. Cependant la plupart des auteurs échouent sur le fait que les opérations booléennes font disparaitre beaucoup de détails. Cette approche ne fonctionne bien que si les primitives sont juxtaposées. Aldefeld **[Aldefeld]** a proposé un très beau travail qui reconnait les primitives du type prisme, couplé avec une aide interactive qui permet de reconstituer de nombreux solides **[Aldefeld 83,84].**

V-1-4 L'approche frontières

On peut considérer le travail de Markowsky et Wesley **[Markowsky & Wesley 80,81,86]** comme la première exposition correcte de l'approche par les frontières. Ils ont donné des définitions mathématiques et un cadre de travail efficace dont nous nous sommes beaucoup inspirés. C'est à eux que nous devons l'idée du fil-de-fer intermédiaire, leur approche souffre peut être cependant de cette rigueur, car voulant trouver toutes les solutions dans tous les cas leurs algorithmes sont souvent peu efficaces. Le travail de Markowsky et Wesley était limité aux polyèdres, Sakurai et Gossard **[Sakurai & Gossard]** l'ont étendu aux surfaces courbes. Cependant leur travail est moins rigoureux et peu

efficace. Nous pensons que nos algorithmes vont plus loin car ils autorisent seulement deux vues, alors qu'il en faut absolument trois chez Sakurai et Gossard, ils trouvent plus de solutions grâce au traitement des arêtes de tangence et des arêtes silhouettes.

V-2 Bilan et perspectives

Nous allons maintenant faire le bilan de ce travail, en commençant par décrire rapidement le programme qui a été réalisé, puis en établissant quels sont les solides reconnus et non reconnus par l'algorithme et enfin en proposant des pistes d'extensions futures.

V-2-1 Programmation

Nous avons mis en oeuvre ces algorithmes pour les valider sur un ordinateur VAX/11 780 sous système VMS. Le système de C.A.O EUCLID a permis de se concentrer uniquement sur la programmation de l'algorithme lui-même puisqu'il fournissait l'entrée des données, l'affichage des résultats et l'archivage. De plus EUCLID est un langage de programmation géométrique qui fournit de nombreuses primitives géométriques que nous avons utilisées.

L'ensemble des programmes que nous avons écrits représente environ cinq mille lignes de FORTRAN. Les dessins présentés en annexe donnent une idée de la complexité atteinte, pour le plus complexe (la pièce MBB) il faut environ deux minutes de temps CPU.

V-2-2 Solides reconnus

Nous pouvons faire le point de ce qui est dit de part et d'autre de ce rapport pour préciser quels sont les solides qui sont reconnus par ce systèmes :

- Au niveau du fil-de-fer tous les solides dont les surfaces sont des plans, cylindres, cones, sphères ou tores et dont toutes les arêtes sont des segments de droite ou des arcs de cercles sont reconnus sans problèmes.

- Au niveau des projections il faut ajouter aux précédentes la condition que les axes des surfaces (cylindre, cone, tore) soient parallèles aux axes de référence.

V-2-3 Extensions

- Plusieurs extensions se présentent que nous pouvons envisager d'intégrer dans une version future de cet algorithme.

- Autre types de courbes : nous pensons particulièrement aux coniques qui permettent de prendre en compte des surfaces d'orientations plus variées, c'est à dire de relacher la contrainte sur les axes . Voire les courbes d'intersections dans toute leur généralité. Une courbe peut alors etre définie par l'intersection des surfaces prismatiques formées par les projections de la courbe et les axes de référence.

- Parties cachées : les projections que nous utilisons sont des projections sans élimination des parties cachées. Prendre en compte les parties cachées demande un effort particulier dans une direction toute nouvelle, puisqu'il s'agit d'imaginer des techniques pour "deviner" où manquent des arêtes et lesquelles ajouter sans être submergé par l'explosion combinatoire tout en ayant de bons résultats. Une approche de type " intelligence artificielle " doit être utilisée pour reconnaitre les configurations les plus courantes pour lesquelles on sait imaginer les parties cachées manquantes.

Bibliographie

[Aldefeld 83] B. Aldefeld On Automatic recognition of 3D structures from 2D representations *Computer-Aided Design* Vol.15 No.2 mar. 1983. pp 59-64

[Aldefeld 84] B. Aldefeld and H. Richter Semiautomatic three-dimensional interpretation of line drawings *Computer & Graphics* Vol.8 No.4 1984. pp 371-380

[Ansaldi & all 85] S. Ansaldi, L. De Floriani, B. Falcidieno Geometric modeling of solid objects by using a face adjacency graph representation *ACM proceedings SIGGRAPH 85 San Francisco* Vol 17 No 3 1985 pp 131-139

[Baumgart 75] B. Baumgart A polyhedron representation for computer vision *AFIPS Proc.,* Vol 44 1975 pp. 589-596.

[Böhm & all 84] W. Böhm, G. Farin, J. Kahmann A survey of curve and surface methods in CAGD *Computer aided geometric design* No 1 1984 pp 1-60

[Bondy & Murty 76] J.A. Bondy, U.S.R. Murty Graph theory with applications *North Holland* 1976

[Braid 75] I.C. Braid The synthesis of solids bounded by many faces *Communications of the ACM* Vol 18 No 4 April 1975 pp 209-216

[Courter & Brewer 86] S.M. Courter, J.A. Brewer Automated conversion of curvilinear wire-frame models to surface boundary models: a topological approach *ACM proceedings SIGGRAPH 86 Dallas* Vol 20 No 4 pp 171-178

[Demoucron & all 64] G.Demoucron, Y.Malgrange, R.Pertuiset Graphes planaires: reconnaissance et construction de représentations planaires topologiques *Revue française de recherche opérationnelle* No 8 pp 33-47

[Eastman & Weiler 79] C. Eastman, K. Weiler Geometric modelling using the Euler operators *1st Annual Conf. Comp. Graphics in CAD/CAM* MIT (1979) pp 248-259

[Edmonds 60] J. Edmonds A combinatorial representation for polyhedral surfaces *American Mathematical Society Notices* Vol.7 Oct 1960 p 646

[EUCLID] EUCLID Manuel de référence langage *Matra-Datavision*

[Faux 87] I.D. Faux Benefits of the parametric evaluator for effective product data exchange and CIM system development *Proceedings of the CAMI 87 Boston* 1987

[Ganter & Uicker 83] M.A. Ganter, J.J. Uicker Jr. From wire-frame to solid-geometric: Automated conversion of data representation *Computers in mechanical engineering* September 1983. pp 40-45

[Goldman 83] R. R. Goldman Two approaches to a computer model for quadric surfaces *IEEE Computer Graphics and Applications* September 1983. pp 21-24

[Grünbaum 67] B. Grünbaum Convex polytopes *Pure and applied matematics* volume XVI, interscience publishers, John WILEY & Sons (1967).

[Hanrahan 82] P.M. Hanrahan Creating volume models from edge-vertex graphs *Computer Graphics* Vol 16 No 3 July 1982 pp 77-84

[Haralick & Queeney 82] R.M. Haralick, D.Q. Queeney Understanding engineering drawings *Computer Graphics and image processing* 20 1982 pp 244-258

[Hilbert] D. Hilbert, D. Cohn-Vossen Geometry and the imagination *Chelsea*

[Hoffman & Hopcroft 87] C.M. Hoffmann, J.E. Hopcroft Geometric ambiguities in boundary representations *Computer aided design* Vol.19 No.3 april 1987. pp 141-147

[Idesawa 73] M. Idesawa A system to generate a solid figure grom three views *Bulletin of the J.S.M.E* Vol.16 No 92 , Febr. 1973 pp 216-225.

[Idesawa 75] M. Idesawa, S. Shibata Automatic input of line drawing and generation of solid figure from three-view data *Proceedings International Computer Symposium* 1975 Vol 2 pp 304-311.

[Jackins & Tanimoto 80] C.L. Jackins and S.L. Tanimoto Oct-trees and their use in representing three-dimensional objects *Computer Graphics and image processing* 14 1980. pp 248-270.

[Kalai 82] Y. Kalai Determinig the spatial containment of a point in general polyhedra *Computer Graphics and image processing* 19 1982. pp 303-334

[Lafue 78] G. Lafue A theorem prover for recognizing 2D representation of 3D objects *Proc. IFIP TC-5 Working. Conf. on Artificial Intelligence and Computer Aided Design* Grenoble (Mars 1978) pp 391-401.

[Lafue 76] G. Lafue Recognition of three-dimensional objects from orthographic views *Proceedings 3rd Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing* ACM SIGGRAPH July 1976 pp 103-108.

[Lakatos 86] I. Lakatos Preuves et réfutations *Herman* 1986

[Lee & Requicha 82] Y.T. Lee, A.A.G. Requicha Algorithms for computing the volume and other integral properties of solids *Communications of the ACM* Vol.25 No.9 September 1982. pp 635-650

[Liewald 85] M.H. Liewald Initial Graphic Exchange Specification : Successes and evolution *Computer & Graphics* Vol.9 No.1 1985. pp 47-50

[Mantyla 82] M. Mantyla, R. Sulonen GWB : a solid modeler with Euler operators*IEEE Computer Graphics and Applications* September 1982. pp 17-31

[Mantyla 84] M. Mantyla A note on the modeling space of Euler operators *Computer vision , graphics, and image processing* 26 1984 pp 45-60

[Markowsky & Wesley 80] G. Markowsky, M.A. Wesley Fleshing out Wire Frames *IBM Journal of research and developement* Vol .24 No.5 Sep. 1980. pp 582-597

[Markowsky & Wesley 81] G. Markowsky, M.A. Wesley Fleshing out Projections *IBM Journal of research and developement* Vol .25 No.6 Nov. 1981. 934-954

[Markowsky & Wesley 86] G. Markowsky, M.A. Wesley Generation of solid models from two-dimensional and three-dimensional data in *Solid modelling by computer : from theory to application* ed M.S. Pickett, J.M. Boyse 1986 Plenum pub. corp. pp 23-51

[Meagher 82] D. Meagher Geometric modeling using octree encoding *Computer Graphics and image processing* 19 1982. pp 129-147

[Preiss 80] K. Preiss Constructing the 3-D Representation of a Plane-faced Object from a digitized Engineering Drawing *Proceedings of linternational Artifificial Intelligence Conference* Brighton, England, April 1980 pp 257-265

[Preiss 81] K.Preiss Algorithms for automatic conversion of a 3-view drawing of a plane-faced part to the 3D representation *Computer Industry* Vol 2 (1981) pp 133-139..

[Preiss 84] K.Preiss Constructing the solid representation from engineering projections *Computers & Graphics* Vol 8 No 4 1984 pp 381-389

[Requicha 80] A.A.G. Requicha Representations for rigid solids : theory, methods, and systems *Computer Surveys* Vol 12 No 4 December 1980. pp 437-464

[Requicha & Voelcker 77] A.A.G. Requicha, H.B. Voelcker Geometric modelling of mechanical parts and processes *Computer* December 1977. pp 48-56

[Requicha & Voelcker 82] A.A.G. Requicha, H.B. Voelcker Solid modelling : A historical summary and contemporary assessment *IEEE Computer Graphics and Applications* March 1982

[Requicha & Voelcker 83] A.A.G. Requicha, H.B. Voelcker Solid modelling : Current status and research directions *IEEE Computer Graphics and Applications* October 1983 pp 25-37

[Requicha & Voelcker 85] A.A.G. Requicha, H.B. Voelcker Boolean operations in solid modelling: boundary evaluation and merging algorithms *Proceedings of the IEEE* Vol 73 No 1 January 1985. pp 30-44

[Rubin 75] F. Rubin An improved algorithm for testing graph planarity *IEEE Transactions on computers* C-24 1975. pp 113-121

[Roth 82] S.D. Roth Ray casting for modeling solids *Computer graphics and image processing* 18 1982. pp 109-143

[Sabatier 84] G.Sabatier Algorithme et programme de calcul de polyèdres convexes à partir de leur graphe associé *Thèse de 3eme cycle Université du Languedoc Montpellier* R.R. n° 16 novembre 1984

[Sakurai & Gossard 83] H. Sakurai, D.C. Gossard Solid model input through orthographic views *Computer Graphics* Vol 17 No 3 July 83. pp 243-252

[Sutherland 74] I.E. Sutherland Three-dimensional data input by tablet *Proceedings of the IEEE,* Vol.62, No 4, April 1974. pp 453-461

[Tiller 83] W. Tiller Rational B-Splines for curves and surface representation *IEEE computer graphics and applications* September 1983. pp 61-69

[Tilove 80] R.B. Tilove Set membership classification : a unified approach to geometric intersection problems *IEE Transactions on computers* Vol.C-29 No.10 October 1980. pp 874-883

[Wang 86] W.P. Wang, K.K. Wang Geometric modeling for swept volume of moving solids *IEEE Computer Graphics and Applications* December1986 pp 8-17

[Weiler 85] K. Weiler Edge-based data structure for solid modeling in curved-surface environments *IEEE Computer Graphics and Applications* January 1985. pp 21-40

[Wesley & all. 80] M.A. Wesley, T; Lozano-Pérez, L.I. Lieberman, M.A. Lavin, D.D. Grossman A geometric modeling system for automated mechanical assembly *IBM Journal of research and developement* Vol .24 No.1 Jan. 1980. pp 64-74

[White 73] A. White IGraphs, groups and surfaces *Mathematical studies 8* North Holland Amsterdam 1973

[[Wilson & all 82] P.R. Wilson, I.D. Faux, M.C. Ostrowsky, K.G. Pasquill Interfaces for data transfer between solid modelling systems *IEEE Computer Graphics and Applications* May 1982

[Wilson 85] P.R. Wilson Euler formulas and geometric modelling *IEEE Computer Graphics and Applications* August 1985 pp 24-36

[Wilson 87] P.R. Wilson Modeling abstractions *Proceedings of the CAM-I Boston* 1987 pp 1-13

[Wilson 87] P.R. Wilson Conic representation for shape description *IEEE Computer Graphics and Applications* April 1987 pp 23-30

[Woo 85] T.C. Woo A combinatorial analysis of boundary data structures schemata*IEEE Computer Graphics and Applications* March 1985 pp 19-27

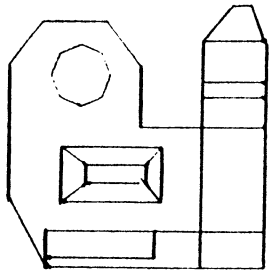
[Yoshiura 84] H. Yoshiura, K. Fujimura, T.L. Kunii Top-down construction of 3-D mechanical objetcs shapes from engineering drawings *Computer* December 1984 pp 32-40

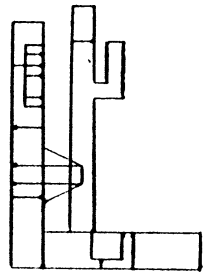
Annexe

Nous présentons maintenant quelques exemples de solides reconstruits avec le programme qui met en oeuvre les algorithmes décrits dans ce rapport.
Les temps CPU sur le VAX 11/785 vont de quelques secondes pour les objets les plus simples à trois minutes pour la pièce la plus compliquée (celle intitulée "benchmark courant en CAO").

Pour les premiers exemples nous avons ajouté le fil-de-fer intermédiaire, ensuite nous illustrons quelques traits particuliers : arêtes de tangences, surfaces toriques, projections ambigues, intersection de prismes. Enfin nous donnons d'autres exemples sans commentaires.

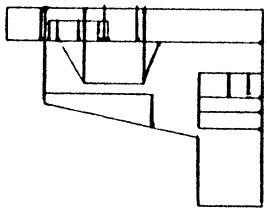
Un solide polyédrique à partir de trois vues

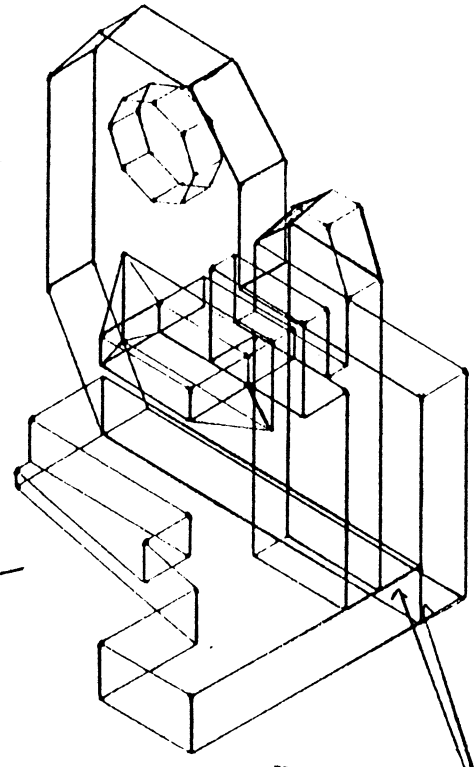




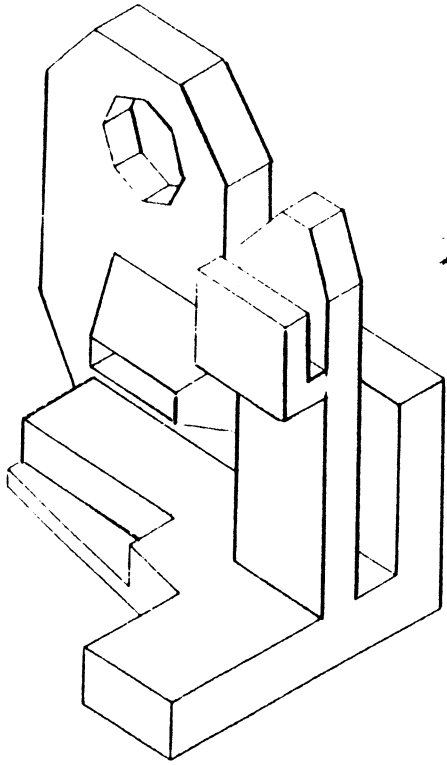
Vues

Fil -de-fer intermédiaire



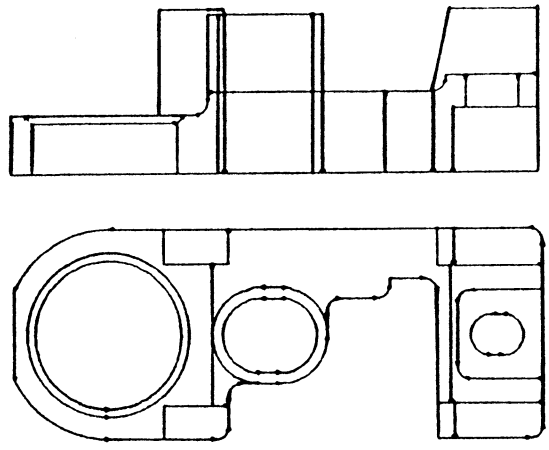


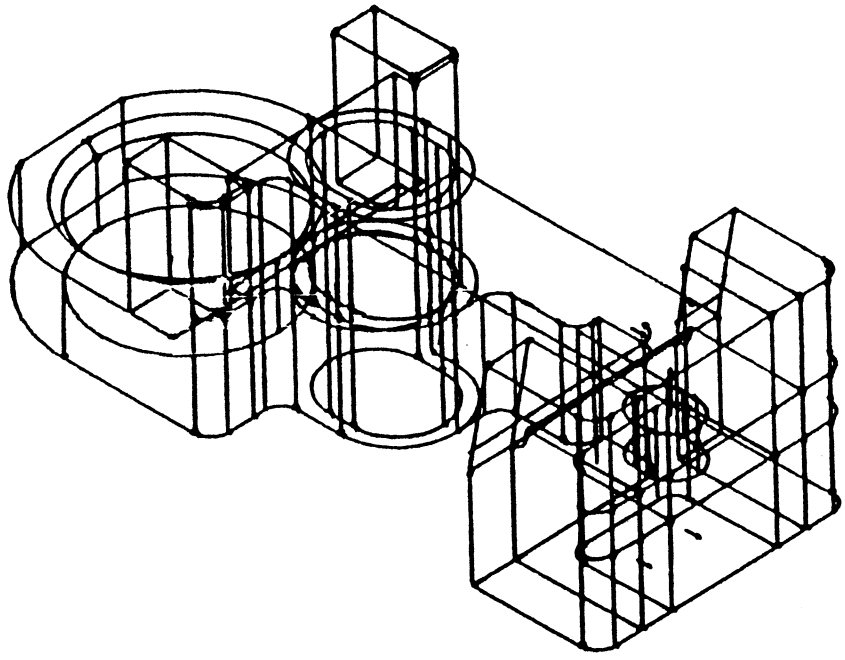
Pseudo-arête

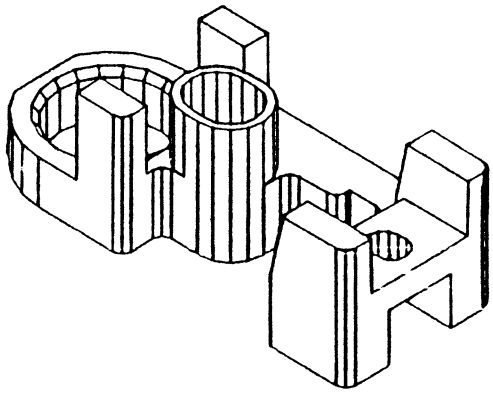


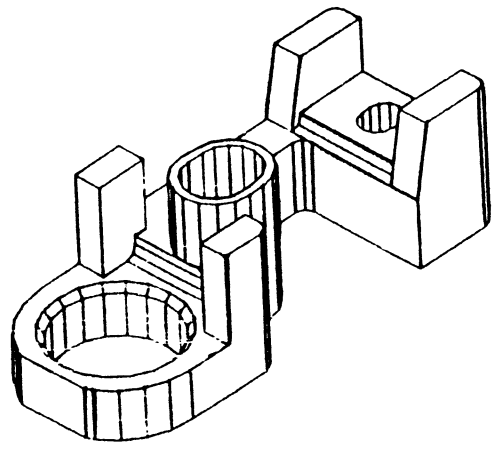
Solide

Un benchmark courant en C.A.O à partir de deux vues :



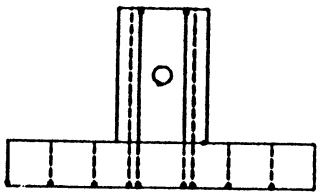




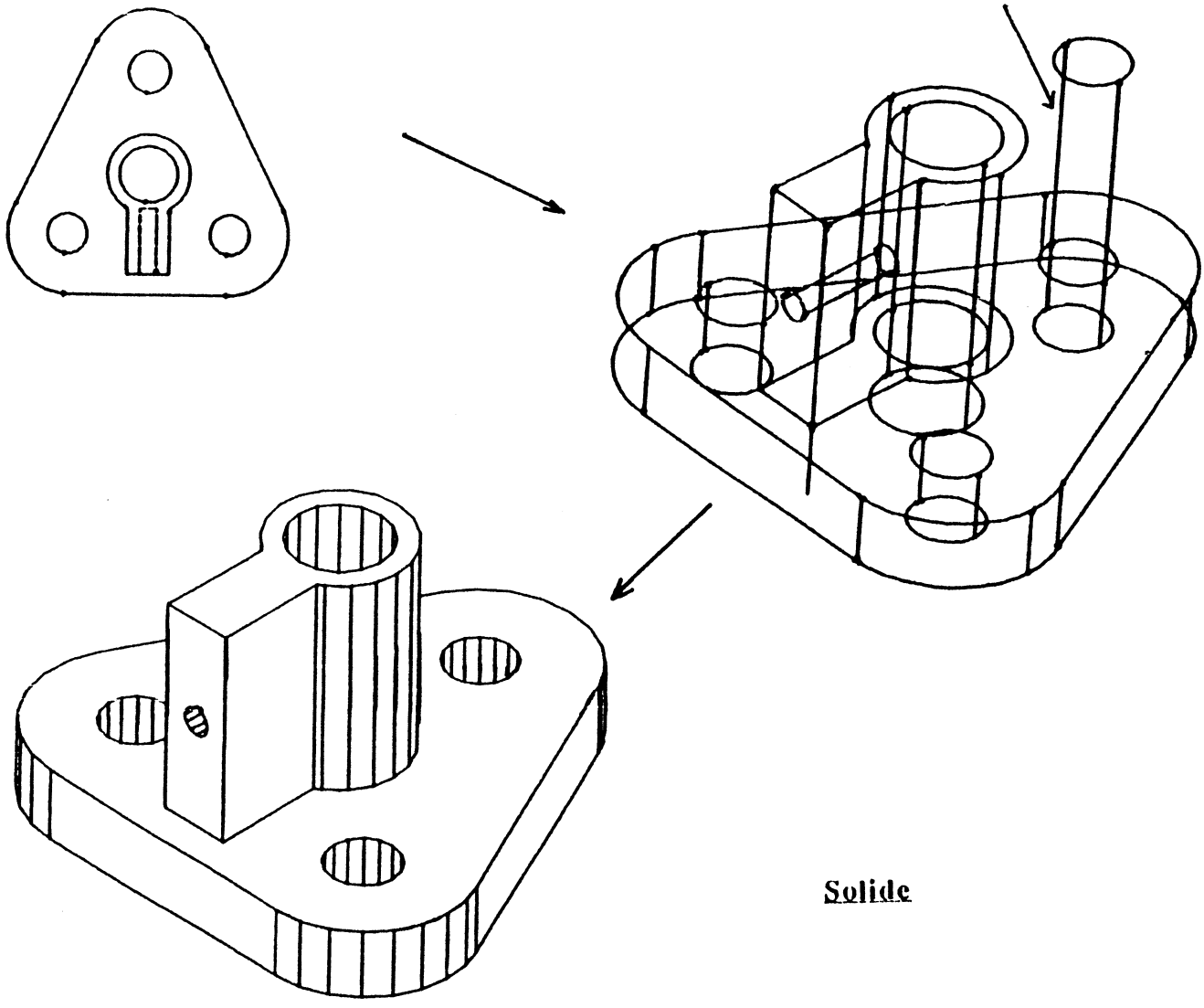


Un solide à partir de deux vues

Vues

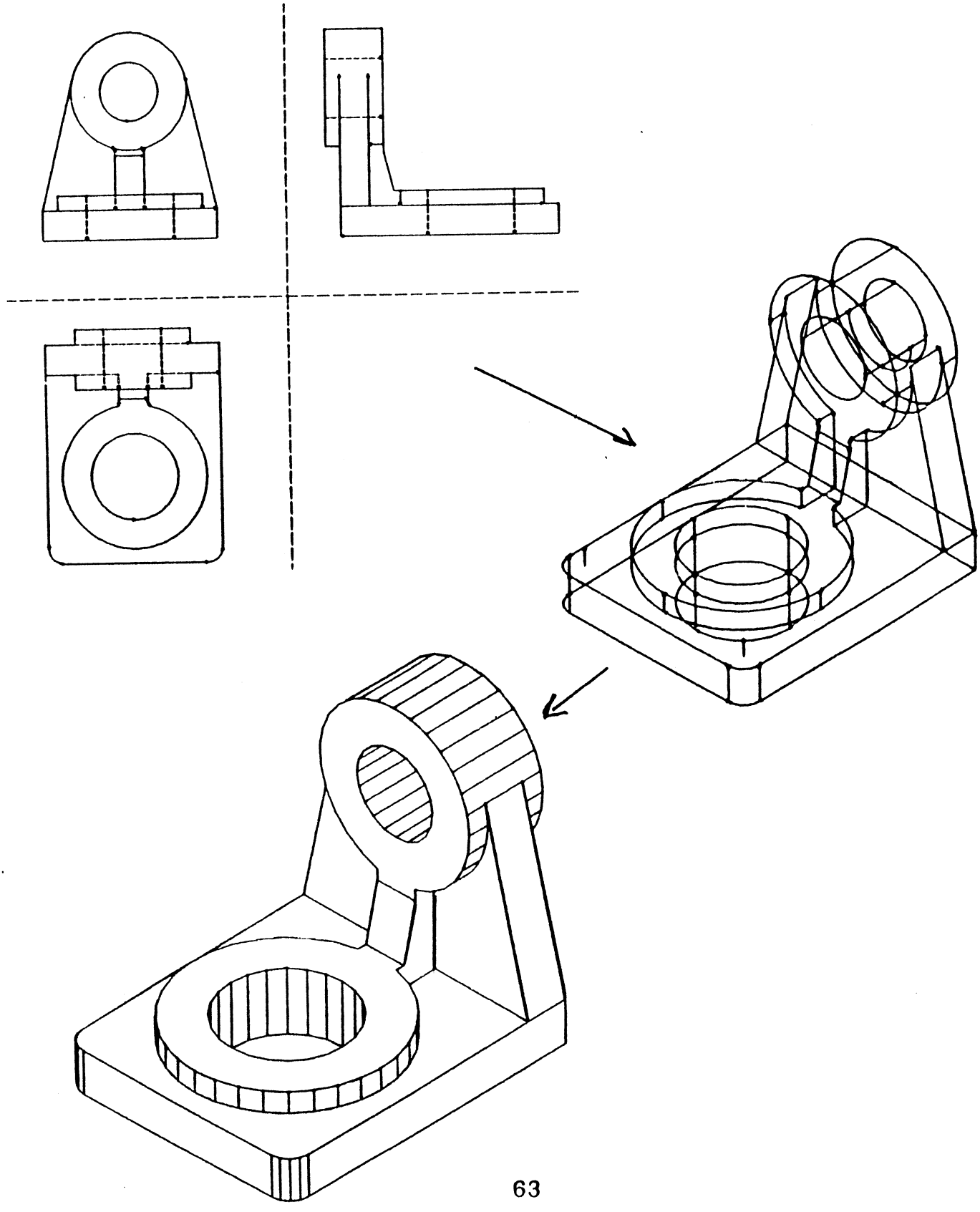


Fil -de-fer intermédiaire



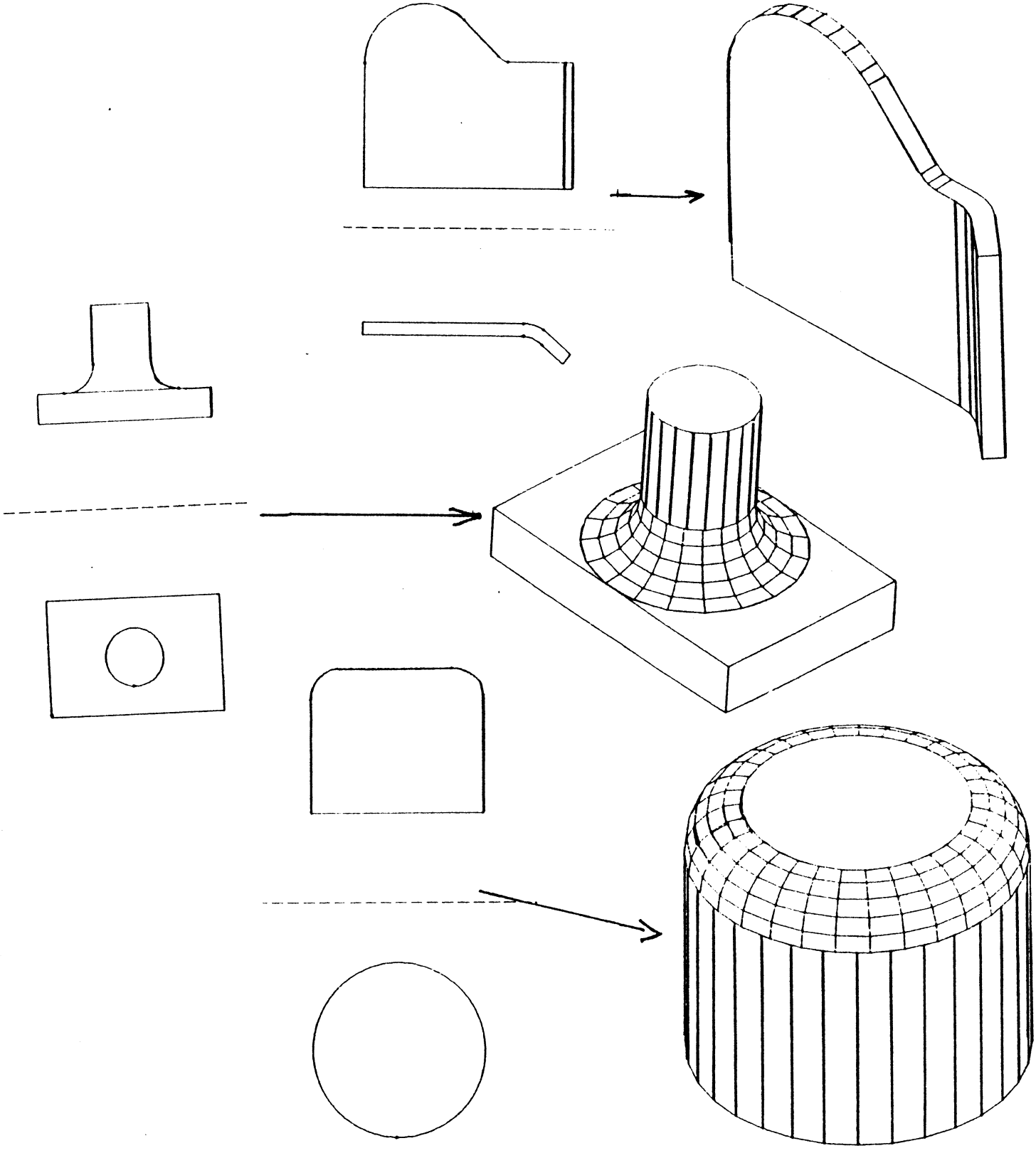
Solide

Une autre pièce mécanique à partir de trois vues



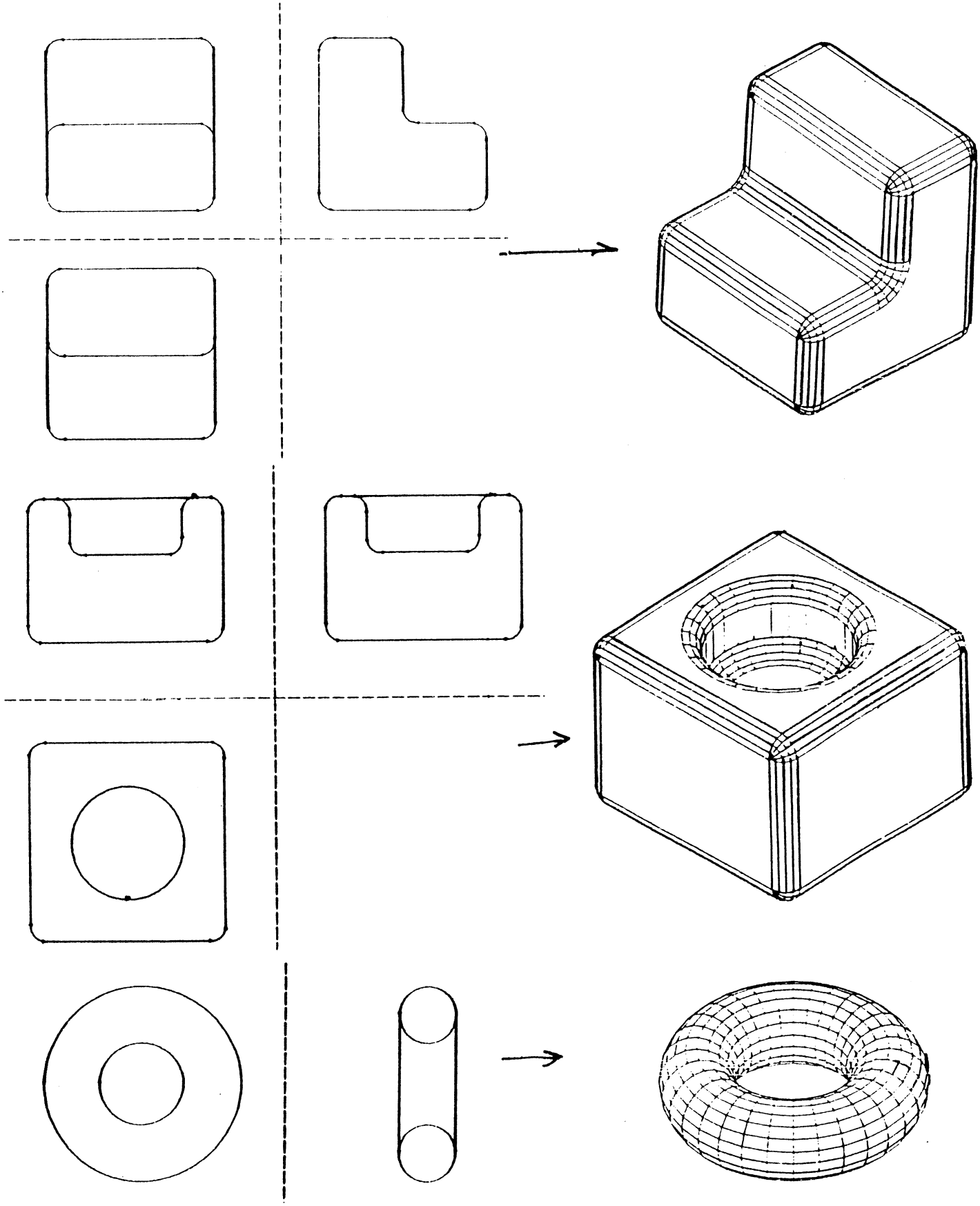
Trois exemples simples mettant en évidence les arêtes de tangence

Remarquer que dans chaque cas deux vues suffisent.



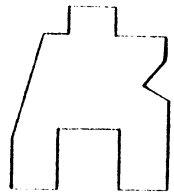
Quelques autres exemples avec des surfaces toriques,

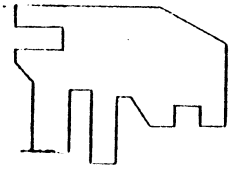
(sans les fil-de-fer intermédiaires)



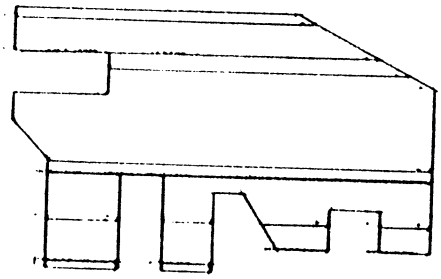
Intersection de deux prismes

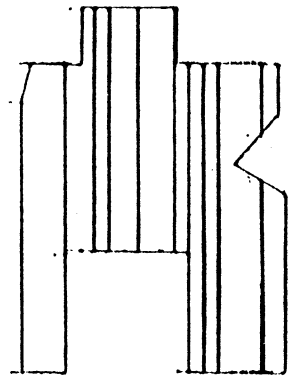
Sections des prismes



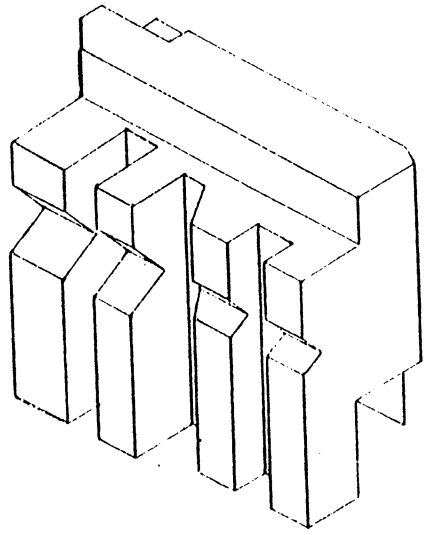


Complément des projections

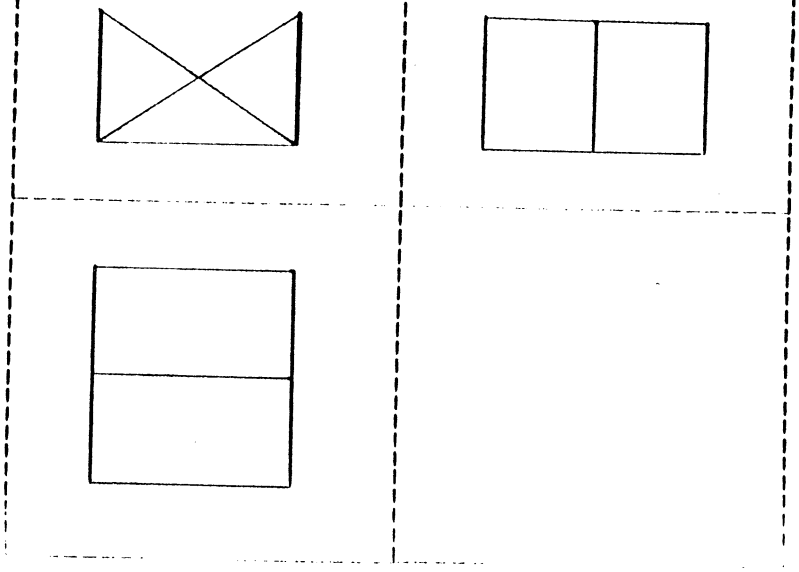


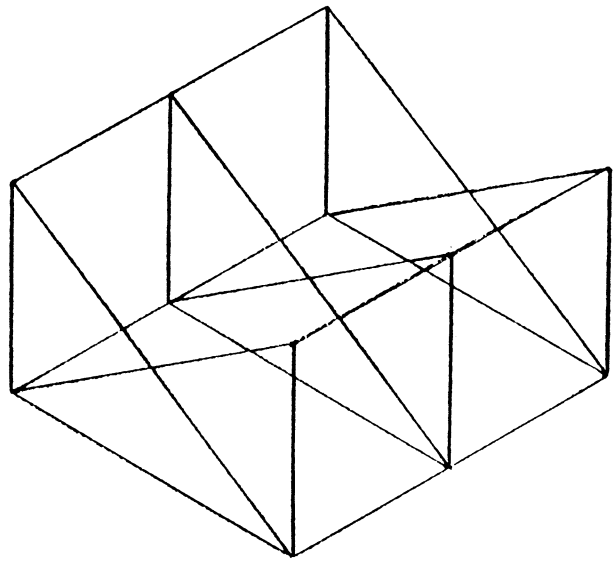


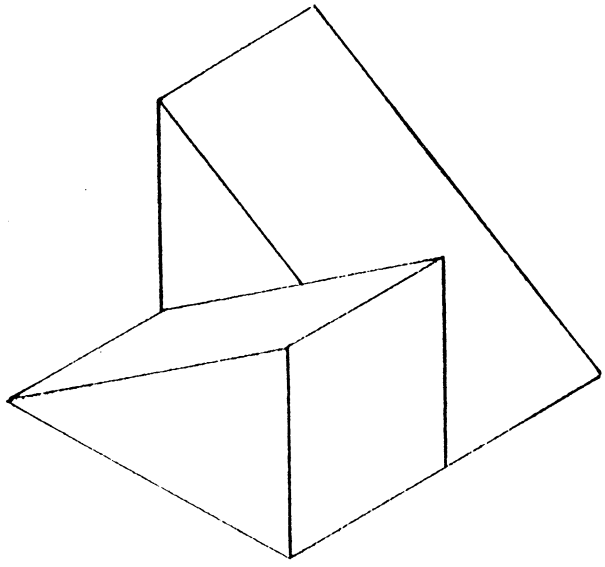
Solide reconstruit

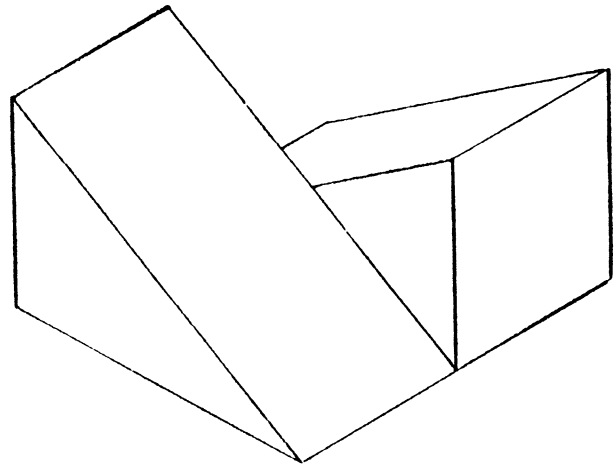


Projections ambigues (deux solutions)

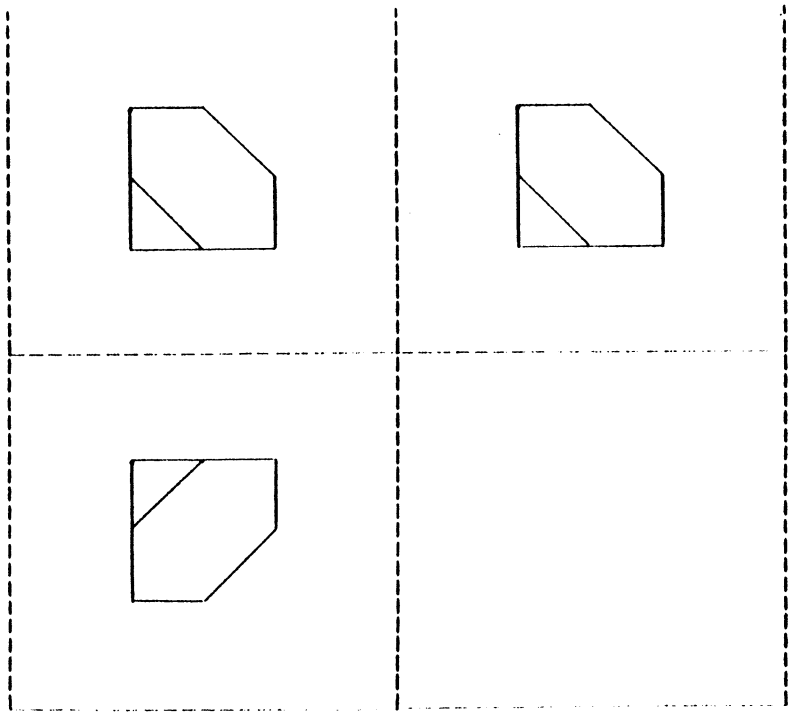


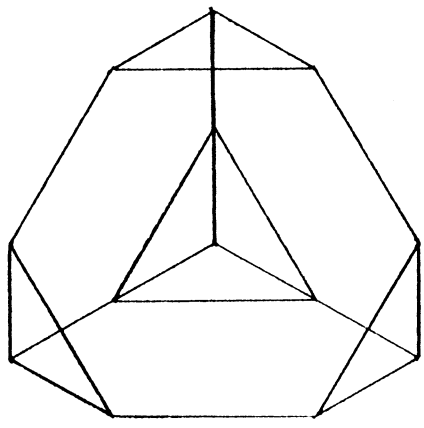


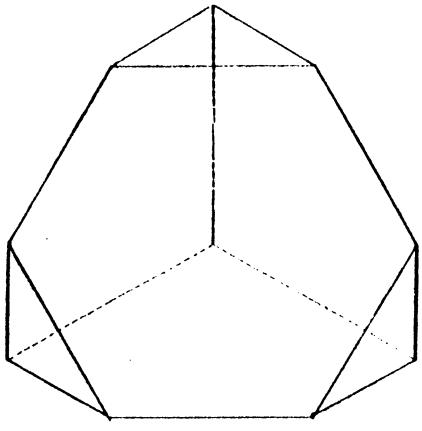


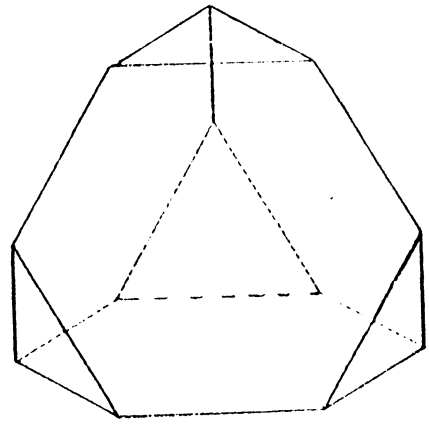


Projections ambigues (deux solutions)

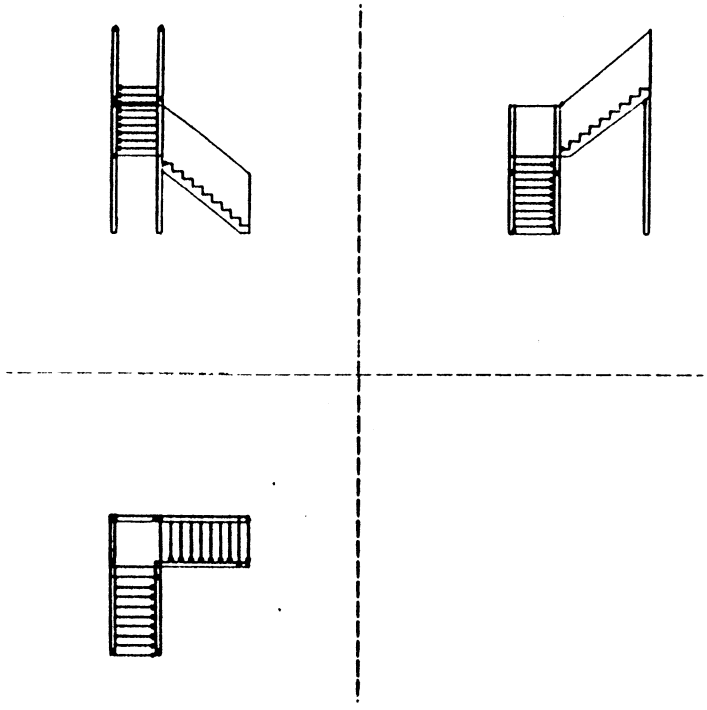


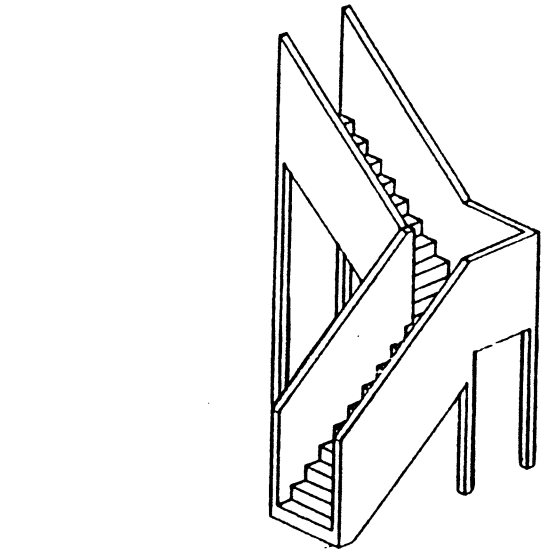


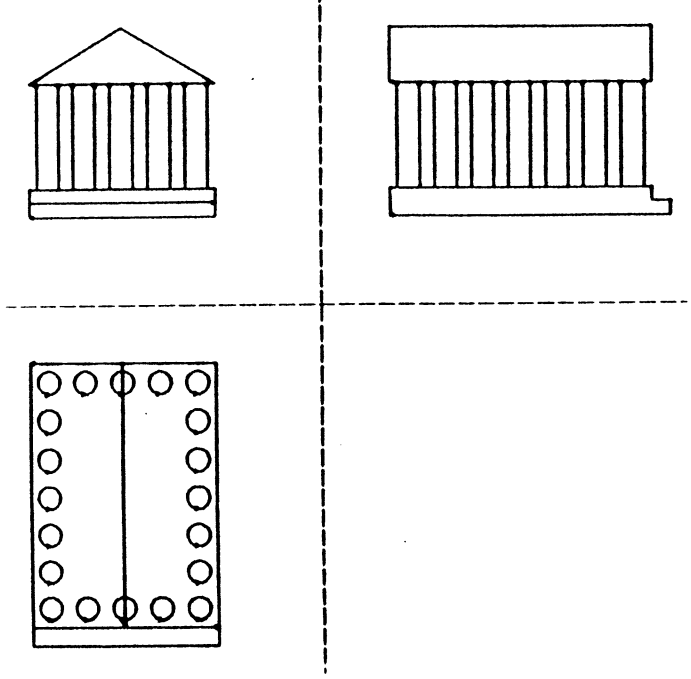


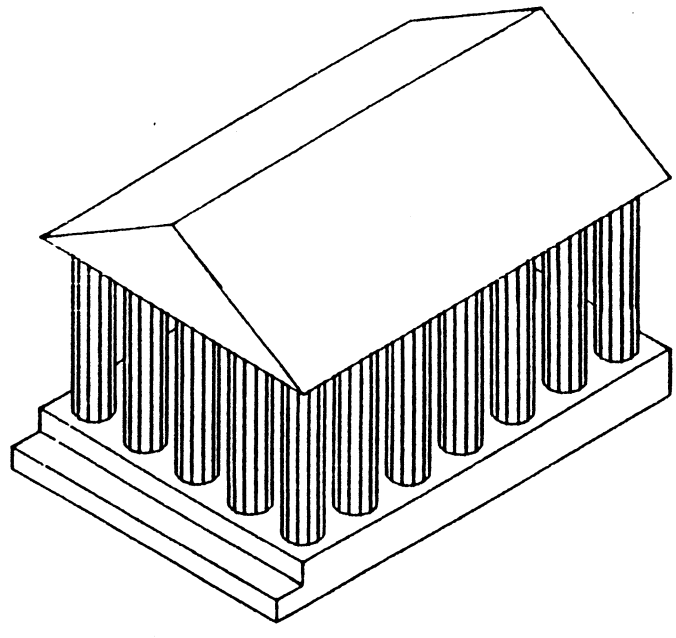


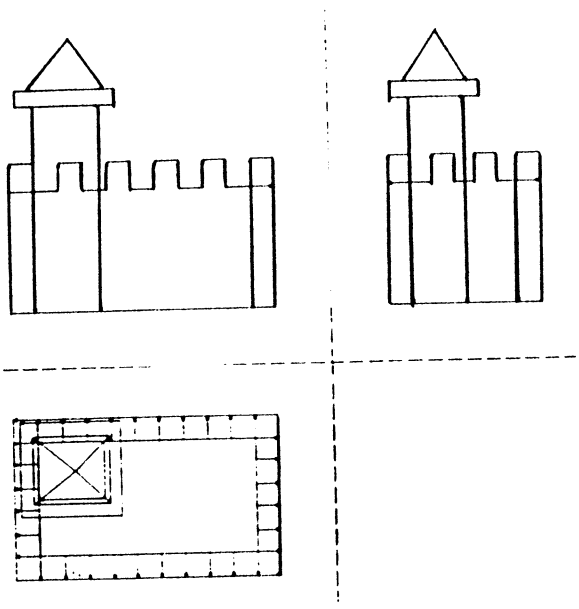
Quelques autres exemples sans commentaires

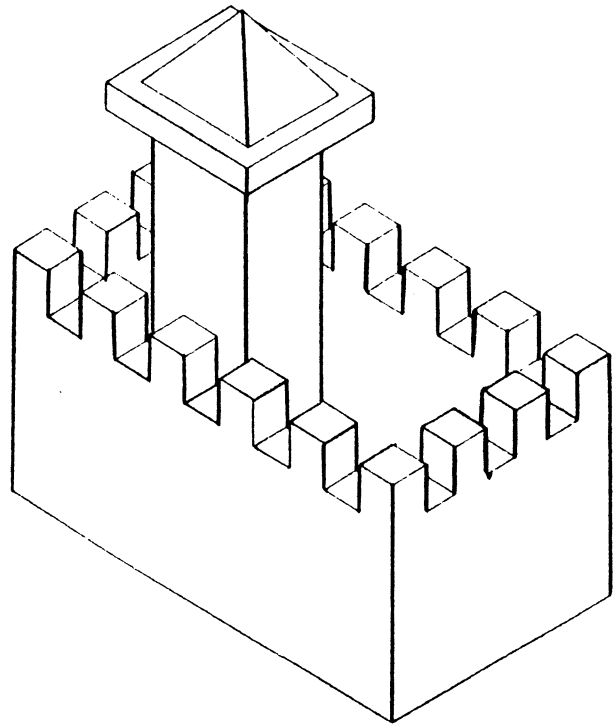


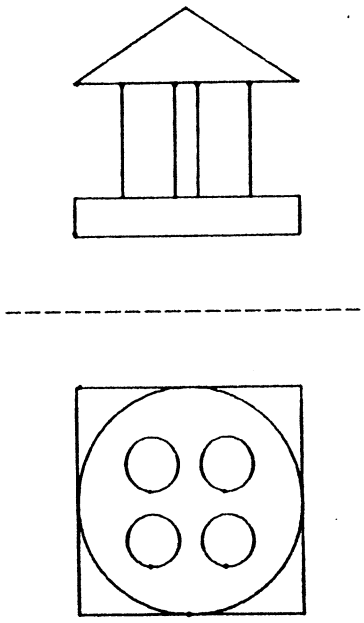


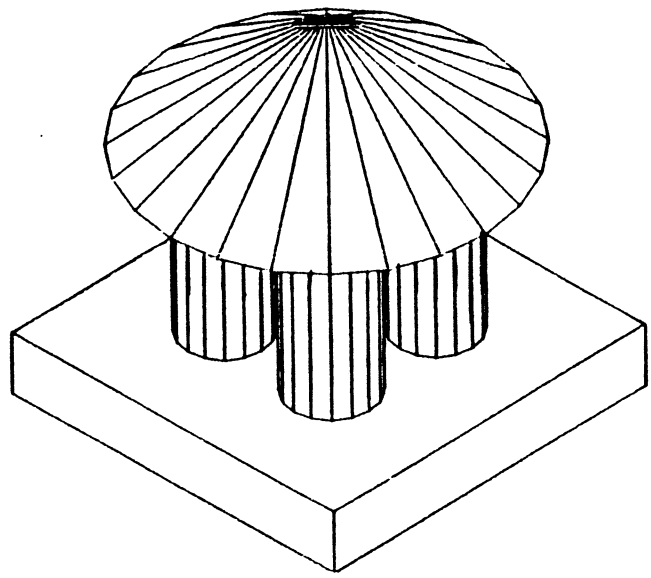


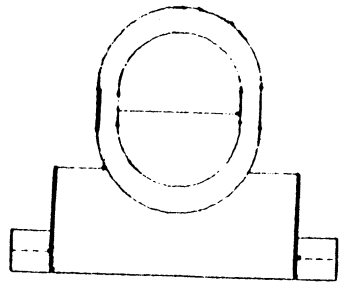


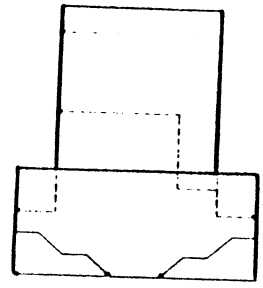


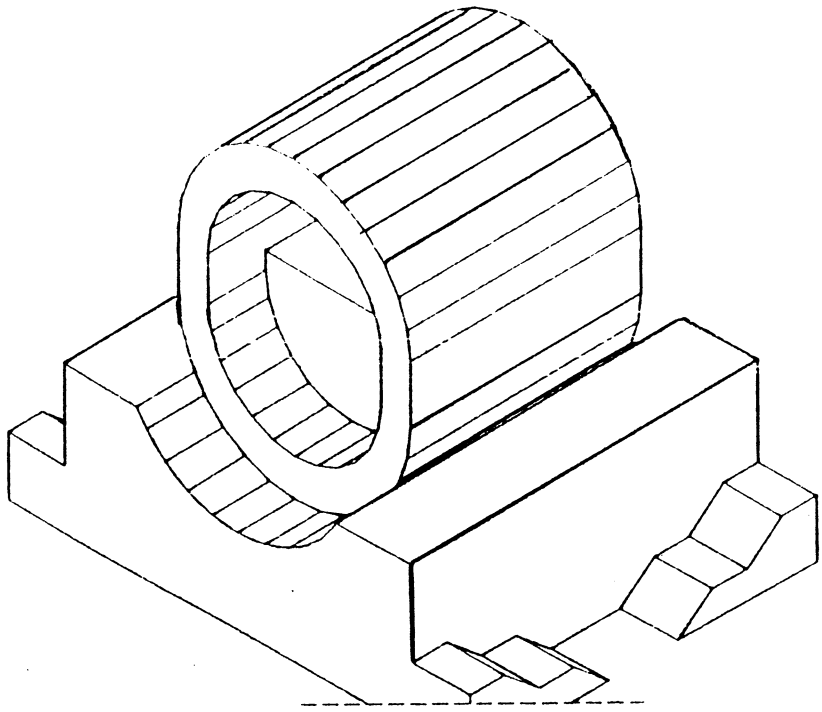


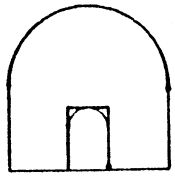


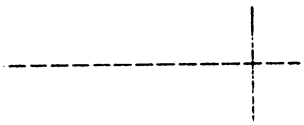


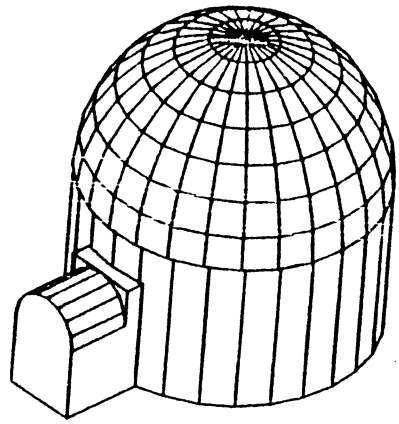


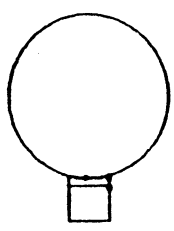


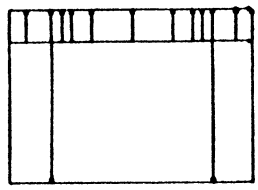


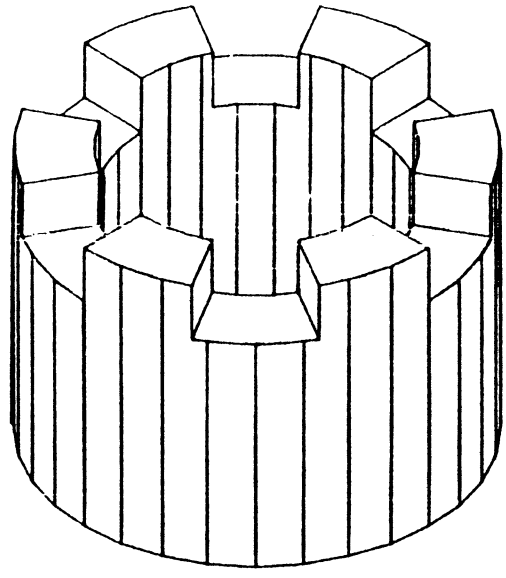


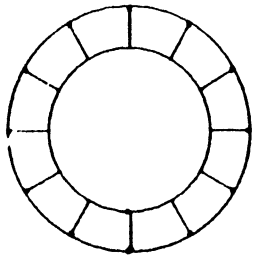


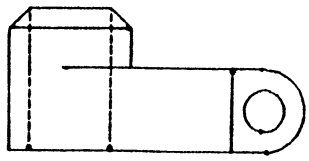


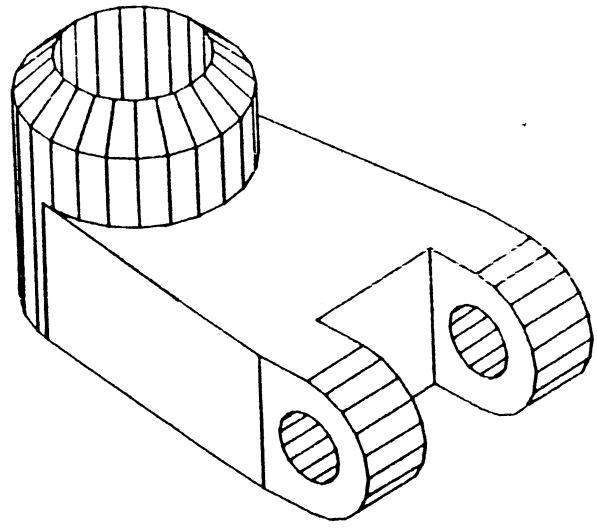


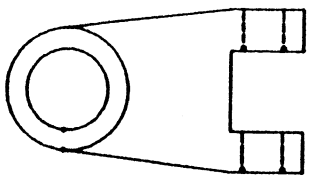


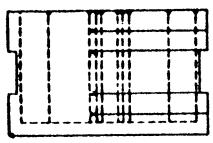


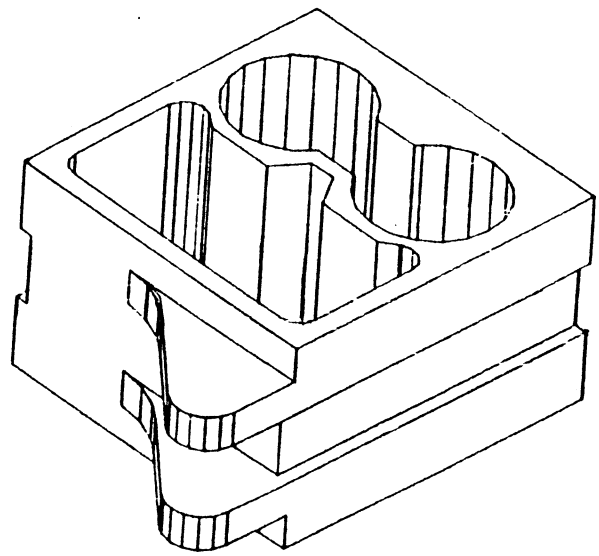


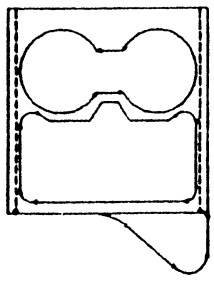












AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, DOCTORAT-INGENIEUR, DOCTORAT USTMG

Vu les dispositions de l'Arrêté du **16 avril 1974**,

Vu les dispositions de l'Arrêté du **5 juillet 1984**,

Vu les rapports de M....LAUBLE.....Bernard.....

M....CRESTIN.....Jean-Pierre.....

M...LEBLETTE....Rémi..... est autorisé

à présenter une thèse en vue de l'obtention du ....Doctorat....U.S.T.M.G—

.....

Grenoble, le.. 20 OCT. 1987 ..

Le Président de l'Université Scientifique Technologique et Médicale

J.J PAYAN