ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PROJET D'APPLICATION PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN TECHNOLOGIE DES SYSTÈMES
M. ING.

PAR
NORMAND GRÉGOIRE

RECONNAISSANCE DE GÉONS DANS UNE DESCRIPTION CAO

MONTRÉAL, DÉCEMBRE 1998

COMPOSITION DU JURY

Ce mémoire a été présenté devant jury le 13 novembre 1998. Le jury était composé des personnes suivantes :

- M. Richard Lepage, directeur de recherche,
  et professeur en génie électrique à l'École de Technologie Supérieure,

- Mme Sylvie Doré, co-directrice de recherche,
  et professeure en génie mécanique à l'École de Technologie Supérieure,

- M. Roland Maranzana,
  professeur en génie de la production automatisée à l'École de Technologie Supérieure,

- M. Tanneguy Redarce,
  maître de conférences à l'Institut National des Sciences Appliquées de Lyon.

RECONNAISSANCE DE GÉONS DANS UNE DESCRIPTION CAO

Normand Grégoire

(Sommaire)

Notre travail s'inscrit dans un contexte d'inspection automatisée d'objets manufacturés. Une difficulté usuelle de ce type d'application consiste à retrouver rapidement dans la base de données le modèle qui correspond à l'objet sous inspection. Cet appariement préliminaire est pourtant incontournable, puisque le contrôle de qualité ne peut s'amorcer que lorsque la référence est clairement identifiée.

Les instigateurs de ce projet se sont donc tournés vers les géons pour tenter d'accélérer ce processus. Proposés par Biederman dans le cadre de la théorie de *Reconnaissance par composantes*, les géons décrivent qualitativement les volumes élémentaires d'un corps ; à ce titre, ils pourraient permettre une identification sommaire mais rapide de l'objet. Notre projet vise donc à identifier les géons des modèles de référence du système, connus par le biais de leur description CAO.

La première étape de notre réflexion consiste à revoir les attributs du géon. Destinés à classifier les géons, il appert en effet que les attributs originaux entraînent souvent une segmentation indue pour des objets manufacturés quelconques. À cet égard, nous proposons l'introduction de géons négatifs pour décrire les cavités, et suggérons certaines simplifications pour les attributs de dimension et de type d'arêtes génératrices. Nous proposons aussi deux possibilités de codification pour la connectivité des géons.

En l'absence d'une définition formelle du géon, et face à la difficulté de formuler mathématiquement une loi de géonisation, nous définissons ensuite chacun des géons à reconnaître au sein des pièces expérimentales. À défaut d'expliciter la logique de découpage, cette démarche permet de définir implicitement la loi de géonisation.

Nous avons implémenté une méthode d'analyse, dite de séparation sur boucle interne. Elle consiste à identifier au préalable les boucles internes du modèle, puis à amorcer un mécanisme d'hypothèse qui recherche un éventuel géon à partir de la boucle interne. Cette prémisse est acceptable, dans la mesure où une boucle interne est toujours le lieu de jonction entre deux ou plusieurs géons. En cas d'échec d'une hypothèse, le

modèle est alors séparé sur la boucle interne, ce qui permet au moins de faire évoluer la connaissance de l'objet.

Avec cette technique, nous parvenons à reconnaître un peu plus de 50 % des géons de toutes les pièces expérimentales. Cependant, on doit souligner que l'éventuel succès de notre algorithme dépend entièrement de la présence de boucles internes ; à ce titre, la même méthode pourrait n'identifier aucun géon dans un autre ensemble de pièces manufacturées.

Nous concluons que la stratégie globale souffre d'un coût d'implémentation élevé, dans la mesure où chaque technique de géonisation concerne un cas géométrique particulier. Nous observons d'autre part qu'il serait peut-être plus approprié pour un système d'inspection automatisée d'admettre plusieurs descriptions géons de chaque objet.

Mots-clés : inspection automatisée, CAO, reconnaissance par composantes, géons

GEON RECOGNITION IN A CAD DESCRIPTION

Normand Grégoire

(Abstract)

This project is concerned with automated inspection of manufactured parts. A common problem in this application is to rapidly find in the database the model corresponding to the inspected object. This matching process is unavoidable, as quality control can only begin when the system clearly knows the reference.

A possible way to speed up the matching is through geons. Initially proposed by Biederman in his theory of Recognition By Components, geons qualitatively describe the primitive volumes of a body ; thus, they could be used for fast indexing of the database, as they shorten an object description. The aim of our project is to recognize the geons of the models used by the system ; analysis starts from the CAD description of these models.

We first examine geons attributes, needed for geon classification. We observe that many manufactured objects would be segmented more than needed with the original attributes. So, we propose negative geons for describing cavities, and suggest some simplifications to the dimension and edge type attributes. We also propose two schemes for coding connectivity between geons.

As geons lack a formal definition, we define next each geons to be recognized among our experimental parts. This approach avoids the pitfall of formulating a mathematical law of geonization, by defining it implicitely through examples.

We implemented an analysis technique based on inner loops. The program first identifies inner loops of the model, and then starts an hypothesis for each inner loop, searching locally for a geon. This is a valid starting point, as inner loops are always where two or more geons meet. When an hypothesis fails, the model is split in two parts on the inner loop, which increases at least the knowledge of the object.

With this technique, our program recognize slightly more than 50 % of the geons of the whole experimental set. We have to mention however that geons can only be extracted when there are inner loops ; with another experimental set, the same algorithm could fail to recognize any geon.

We conclude that the global strategy is costly to implement, as many geonization techniques will be required, each of them addressing a particular geometric situation. We also observe that automated inspection should maybe admit many descriptions for the same object.

Keywords : automated inspection, CAD, recognition by components, geons

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Tableau Page

LISTE DES FIGURES

Figure Page

LISTE DES ABRÉVIATIONS ET DES SIGLES

ASVP	*Alternating Sum of Volumes with Partitioning*
Brep	Représentation par les limites (*Boundary REPresentation*)
CAO	Conception Assistée par Ordinateur
CAPP	Planification de procédés assistée par ordinateur (*Computer Assisted Process Planning*)
CIM	*Computer Integrated Manufacturing*
GC	Cylindre généralisé (*Generalized Cylinder*)
LIVIA	Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle
MLP	Perceptron multi-couches (*Multi Layer Perceptron*)
NURBS	B-spline rationel non-uniforme (*Non-Uniform Rational B-Splines*)
RBC	Reconnaissance par composantes (*Recognition By Components*)
STEP	*STandard for Exchange of Product data*

INTRODUCTION

0.1 Contexte du projet

Depuis la dernière décennie, plusieurs tendances convergentes ont modifié le paysage économique mondial : naissance de nouveaux marchés communs, ou élargissement des espaces économiques existants ; déréglementation de plusieurs secteurs industriels ; réduction ou abolition des tarifs douaniers et des subventions aux entreprises suite aux accords du GATT ; accroissement des possibilités de communication planétaire grâce à Internet ; contraintes de qualité manufacturière minimale imposées de facto par l'adoption des normes ISO 9000 ; etc. Ces différents bouleversements ont engendré ce qu'il est convenu d'appeler la *mondialisation des marchés*, phénomène par lequel les entreprises, quelle que soit leur taille, doivent maintenant affronter une concurrence accrue, même sur les marchés qui leurs étaient traditionnellement réservés.

Cette concurrence, conjuguée aux facteurs évoqués précédemment, impose donc aux entreprises de produire plus et mieux, avec moins : les impératifs de productivité, de compétitivité et de qualité résument clairement cette exigence. À cet égard, il n'est pas surprenant que l'automatisation des procédés soit de plus en plus sollicitée par le secteur manufacturier.

Le travail présenté dans ce rapport constitue justement un fragment d'un projet de recherche plus global, concernant l'inspection automatisée d'objets manufacturés. Ce

dernier projet, s'étalant sur plusieurs années, est issu de la collaboration entre trois partenaires universitaires, à savoir :

- le Laboratoire d'Automatique Industrielle (LAI),
  de l'Institut National des Sciences Appliquées (INSA), de Lyon ;
- le Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
  de l'École de Technologie Supérieure (ETS), de Montréal ;
- l'Institut des Technologies de l'Information (IIT),
  du Conseil National de la Recherche du Canada (CNRC), d'Ottawa.

Dans le cadre de ce projet, les trois laboratoires en question se sont donné pour mandat d'étudier – et autant que possible, d'apporter des éléments de solution – à certains problèmes que pose l'inspection automatique d'objets manufacturés, et qui retardent la pénétration de cette technologie en milieu industriel. Le projet proposé par ces laboratoires est illustré à la figure 0.1.

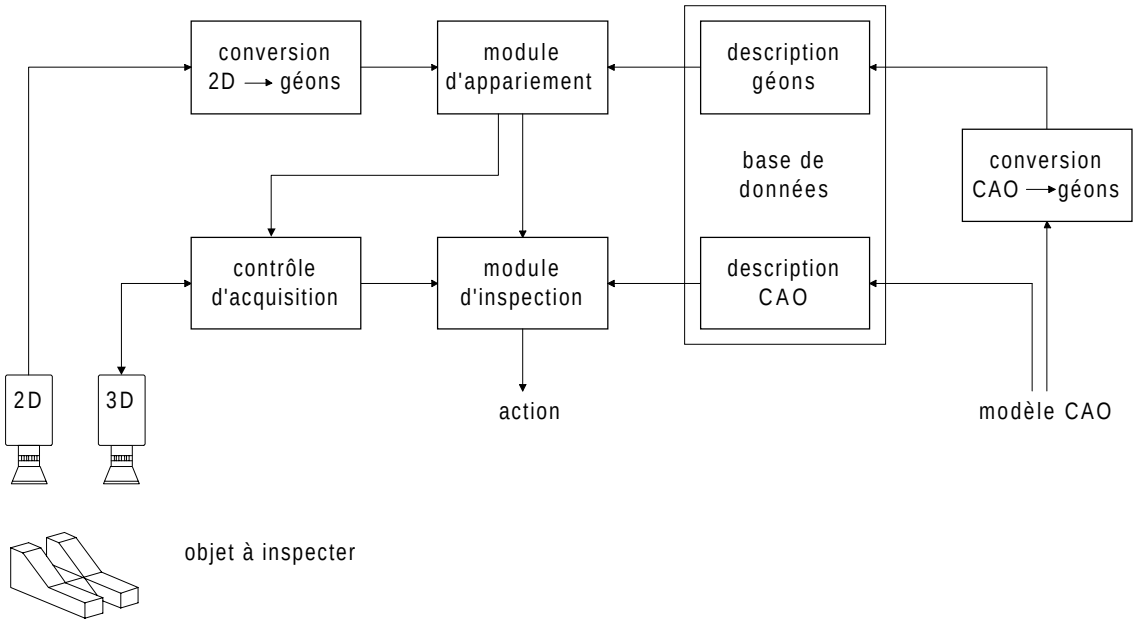


Figure 0.1 : Projet proposé de métrologie industrielle.

De façon générale, l'inspection automatisée d'une chaîne de production doit permettre à un ordinateur de décider si une pièce donnée satisfait aux exigences fixées lors de la conception. Les instruments métrologiques traditionnels impliquent cependant un contact avec la pièce ; afin d'accélérer le processus métrologique, un système de vision est donc proposé pour mesurer les dimensions de la pièce, dimensions qui seront alors confrontées avec la description informatique de l'objet. Une difficulté commune rencontrée à cette étape est celle de la présélection du bon modèle dans la base de données : plusieurs objets différents peuvent transiter simultanément sur la même chaîne, ce qui force l'ordinateur à explorer, dans le pire des cas, la totalité de sa base de données avant de pouvoir commencer l'activité proprement dite de contrôle de qualité.

Dans l'état actuel de la technologie logicielle, le temps requis par cette fouille préliminaire de la base de données s'avère totalement incompatible avec les cadences usuelles de production. Les instigateurs de ce projet se sont donc tournés vers les *géons*, introduits plus loin, afin d'étudier la viabilité de ce concept dans le cadre d'une activité industrielle de métrologie.

0.2 Objectifs généraux et spécifiques

Le projet décrit dans ce rapport fait partie du bloc situé à l'extrême-droite de la figure 0.1. Son objectif général est d'identifier les volumes élémentaires qui définissent un objet donné, et de caractériser leur position relative. Sa finalité est de convertir une description géométrique en une description volumétrique structurelle, c'est-à-dire de transformer une information ample et détaillée sous une forme moins précise, mais plus compacte.

En pratique, un tel objectif ne peut être mené à terme dans le cadre d'un seul projet de maîtrise : en effet, il y a déjà près d'une vingtaine d'années que différentes équipes de recherche travaillent sur ce problème sans qu'une solution globalement satisfaisante n'ait encore émergée. Aussi, notre projet vise t-il à mettre en route le

module de conversion CAO→géons, que d'autres étudiants devront cependant compléter par la suite si les résultats obtenus sont suffisamment prometteurs.

Dans cette optique, nous nous sommes fixés au début de ce projet trois objectifs spécifiques à atteindre :

1. Convertir les données brutes en une structure de données propice à l'analyse géométrique ;

2. Implémenter un algorithme qui, à défaut de pouvoir identifier la totalité des géons, puisse au moins en reconnaître un nombre appréciable ;

3. Développer un outil de visualisation des géons.

Ces trois objectifs représentent en fait chacun des programmes requis pour le traitement et la validation des données, tel qu'illustré à la figure 0.2.

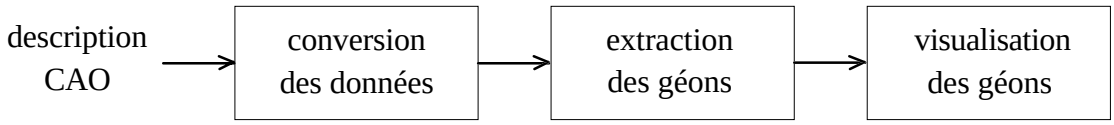


Figure 0.2 : Flux d'information.

0.3 Contraintes générales

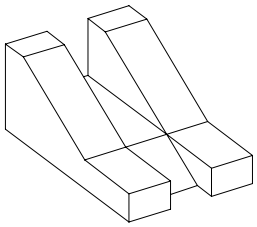
Ce projet doit satisfaire un certain nombre de contraintes ou spécifications, d'autant plus qu'il doit s'intégrer à un système plus complexe. Nous mentionnons ici les prémisses les plus générales du projet :

1. Le module d'extraction de géons doit opérer de façon entièrement automatique, c'est-à-dire sans intervention humaine. Cette contrainte est ambitieuse, dans la mesure où

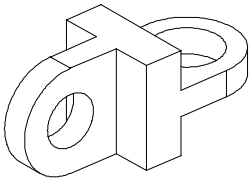
beaucoup de travaux antérieurs demandaient à l'utilisateur de fournir un ou plusieurs indices de départ.

2. Le module doit opérer hors-ligne, c'est-à-dire lorsque le système d'inspection est en phase d'apprentissage plutôt qu'en opération normale. En pratique, cette spécification permet plus de latitude dans le développement des algorithmes, puisque notre projet n'est pas soumis à une contrainte de temps réel.

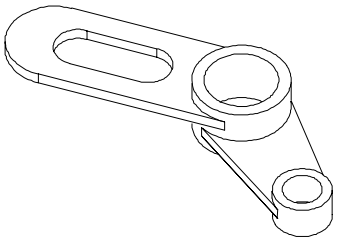
3. Le domaine d'entrée du projet est limité à un univers de dix pièces, illustrées à la figure 0.3. Du point de vue de l'inspection automatisée, il s'agit d'un échantillon très restreint, puisqu'une véritable base de données CAO contiendrait normalement beaucoup plus de modèles. On doit cependant souligner à cet égard que le système proposé est destiné à la validation du concept de géon plutôt qu'à la mise en application d'une théorie éprouvée. En terme de développement, ce petit nombre de pièces permet aussi de mieux orienter le travail, puisqu'il s'avère très laborieux de mettre au point des algorithmes capables de réagir correctement à n'importe quel stimulus.



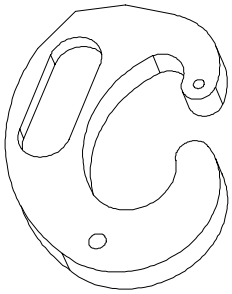
pièce 1



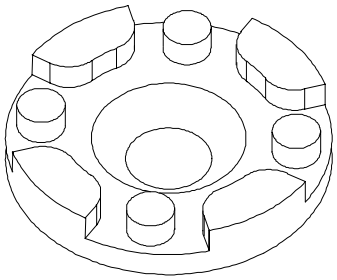
pièce 2



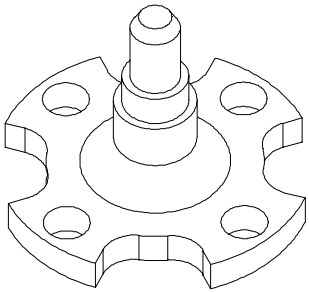
pièce 3



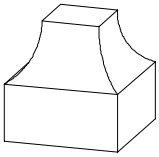
pièce 4



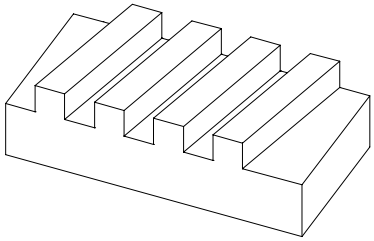
pièce 5



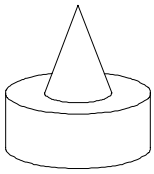
pièce 6



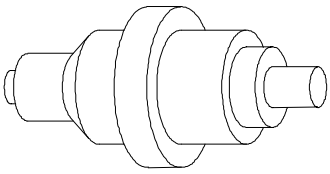
pièce 7



pièce 8



pièce 9



pièce 10

Figure 0.3 : Pièces à reconnaître

0.4 Organisation du mémoire

Ce mémoire comprend six chapitres, divisés tel que suit. Nous exposons au chapitre 1 quelques notions fondamentales pour la compréhension du sujet, dont la théorie des géons. Nous terminons ce chapitre par un exposé sur la modélisation, qui vise surtout à mettre en évidence la forme de nos données d'entrée.

Le chapitre 2 constitue un survol des travaux antérieurs dans le domaine de l'analyse de données CAO. Nous y exposons les tentatives les plus marquantes, en regroupant les différentes approches par famille.

Le chapitre 3 présente la méthodologie retenue pour la réalisation du projet. Le lecteur pourra suivre le cheminement qui nous a guidé dans ce projet, depuis la réflexion théorique sur les géons jusqu'au choix d'un algorithme d'extraction de géons.

Nous présentons au chapitre 4 les différents programmes que nous avons développés, depuis le prétraitement de la description CAO jusqu'à la reconnaissance de géons. Nous discutons de quelques algorithmes nécessaires au raisonnement géométrique, et exposons les différentes prémisses ou contraintes géométriques qu'ils supposent.

Le chapitre 5 présente les résultats obtenus, et discute de certaines faiblesses de l'implémentation actuelle. À la lumière des résultats obtenus, le chapitre 6 propose quelques avenues pour la poursuite du module d'extraction de géons. La conclusion clôt le mémoire par un rappel des éléments marquants de ce projet.

CHAPITRE 1

REPRÉSENTATION D'OBJETS 3D

Avant d'approfondir l'étude du problème qui nous intéresse plus particulièrement, il sera opportun de rappeler certains concepts liés au sujet, et de préciser la terminologie utilisée dans cet ouvrage.

1.1 Cylindres généralisés

Nous présentons dès maintenant le concept de *cylindre généralisé* (GC, pour *Generalized Cylinder*), auquel nous nous référerons souvent par la suite.

Introduits par Binford [1] en 1971, les cylindres généralisés sont maintenant d'usage très courant dans le domaine de la CAO et de l'infographie. Aussi dits *cônes généralisés*, ils réfèrent à la famille des solides engendrés par le déplacement d'une surface le long d'un axe défini dans l'espace 3D ; la figure 1.1 en illustre le principe.

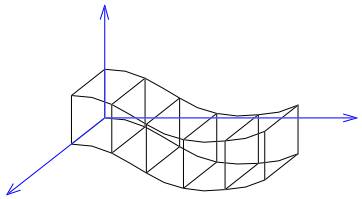


Figure 1.1 : Cylindre généralisé (Adapté de Rogers et Adams [2])

Un intérêt des cylindres généralisés réside dans la possibilité de faire varier la dimension de la surface génératrice : par exemple, un cône peut être modélisé par un GC de section circulaire, et dont la dimension varie linéairement lors du déplacement le long de l'axe.

Dans le cas le plus général, quatre fonctions sont requises pour définir un cylindre généralisé quelconque :

1. La fonction de profil : équation de la surface génératrice. Cette surface étant plane dans la presque totalité des applications, les contraintes sur son domaine seront l'élément essentiel, puisqu'elles définiront la forme du profil générateur. La plupart des applications imposent une contrainte d'*homogénéité*, c'est-à-dire que la forme du profil générateur demeure constante d'une extrémité à l'autre du cylindre généralisé.

2. La fonction axiale : équation de l'axe dans l'espace 3D. Cet axe est le plus souvent rectiligne, ou sinon de courbure régulière. Une description mathématique complète d'un cylindre généralisé impliquerait une paramétrisation de toutes les fonctions par rapport au déplacement le long de l'axe.

3. La fonction de balayage : équation liant la dimension du profil générateur à la position sur l'axe. Cette fonction est généralement constante ou linéaire, bien que certains environnements permettent de définir une fonction de balayage plus complexe. Contrainte importante aux fins de notre projet, la fonction de balayage affecte par un même facteur d'échelle tous les points de contrôle du profil générateur. Par exemple, la surface carrée à l'avant de l'objet illustré à la figure 1.2 ne pourrait agir à titre de profil générateur, puisqu'elle croît selon une seule dimension ; le trapèze latéral satisfait par contre la contrainte d'une mise à l'échelle similaire sur toutes les dimensions.

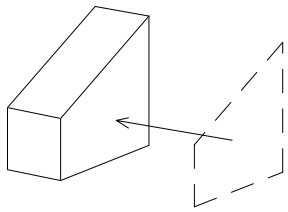


Figure 1.2 : Profil générateur.

4. La fonction d'inclinaison : équation qui exprime l'angle d'inclinaison de la surface génératrice selon son déplacement sur l'axe. La plupart des auteurs omettent cette fonction, puisqu'on impose invariablement que la surface génératrice soit perpendiculaire à l'axe. Sans ce genre de contrainte, on pourrait considérer par exemple que l'objet de la figure 1.2 est produit par balayage du rectangle supérieur le long d'un axe vertical, avec une inclinaison décroissante de la surface génératrice à mesure qu'elle se déplace dans l'espace.

Notre lecteur ne doit d'ailleurs pas se laisser confondre par notre propos : en soi, le cylindre généralisé n'est pas une représentation mathématique des objets, mais seulement une contrainte spatiale imposée aux objets afin de délimiter un univers géométrique.

1.2 Théorie des géons

1.2.1 Reconnaissance par composantes

Dans la foulée du principe de *Recognition by parts* proposé par Binford [1], Biederman formulait dans les années 80 sa théorie de *Reconnaissance par composantes* (RBC, pour *Recognition By Components*), par laquelle il tentait d'expliquer comment une scène complexe est analysée à prime abord par le système visuel humain [3, 4]. Tel que son nom le laisse entendre, la reconnaissance par composantes présume que les objets présents dans le champ de vision sont découpés mentalement selon leurs parties

naturelles, nommément les *géons* ; cet assemblage grossier de volumes élémentaires servirait de clé pour l'identification sommaire de l'objet. La simplicité du concept a incité plusieurs chercheurs en vision à reprendre cette théorie dans un contexte de reconnaissance d'objets 3D dans une image [5–15].

Fondamentalement, un géon pourrait avoir n'importe quelle forme 3D ; cependant, Biederman restreignait les géons à la famille des cylindres généralisés afin de rendre sa théorie plus plausible. Ce choix a d'ailleurs été maintenu dans la plupart des applications de vision artificielle, puisque les GC permettent de modéliser bon nombre d'objets courants, naturels ou manufacturés, tout en limitant le degré de difficulté inhérent au problème.

Malgré les diverses représentations possibles d'un GC, Biederman retenait seulement quatre attributs symboliques pour caractériser un géon, dont les différentes possibilités sont illustrées à la figure 1.3 :

- Axe : droit ou recourbé ;
- Section : – arêtes : droites ou recourbées ;
  – symétrie : réflective, rotationnelle et réflective, ou asymétrique ;
  – dimension : constante, en expansion, ou en expansion et contraction.

Ce choix était d'ailleurs conséquent avec le principe d'une reconnaissance de prime abord (*primal access*), qui excluait toute métrique complexe dans l'opération mentale de découpage. Il n'en demeure pas moins qu'un objet réel est généralement constitué de plusieurs géons : aussi un attribut de connectivité doit-il être spécifié pour chaque paire de géon, de sorte à traduire la position relative des géons. Dans ses travaux, Biederman postulait trois types de relations spatiales, certaines d'ordre géométrique, et d'autres d'ordre perceptuel :

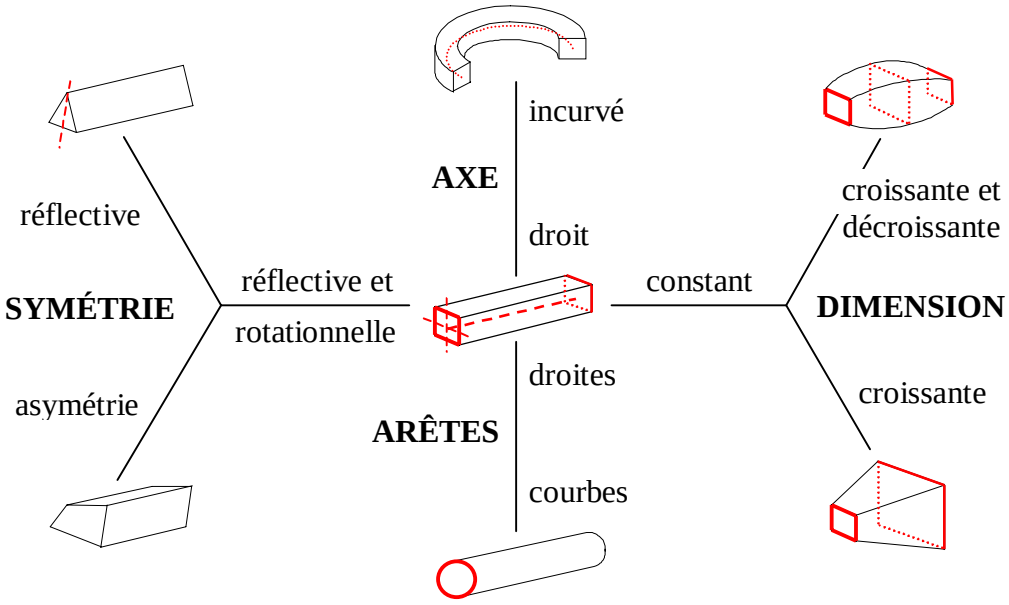


Figure 1.3 : Attributs du géon. (Adapté de [3])

- Type de connexion : bout-à-bout ou par le côté ;
- Point d'attache : à une surface courte ou longue ;
- Position relative : au-dessus ou en-dessous ;

La théorie de *Reconnaissance par composantes* cadre bien avec l'objectif de notre projet : quelques ajustements seront cependant requis pour l'adapter au problème étudié. Ces modifications à la théorie initiale seront exposées au chapitre 3.

1.2.2 Description par géons

La représentation par géons est une description structurelle des objets, généralement codifiée sous forme de graphe relationnel. À chaque géon est associé un noeud du graphe, caractérisé par les attributs intrinsèques du géon ; les arcs, dirigés ou non, expriment les relations d'adjacence entre les différents géons. La figure 1.4 illustre un exemple de graphe relationnel pour une des pièces du projet.

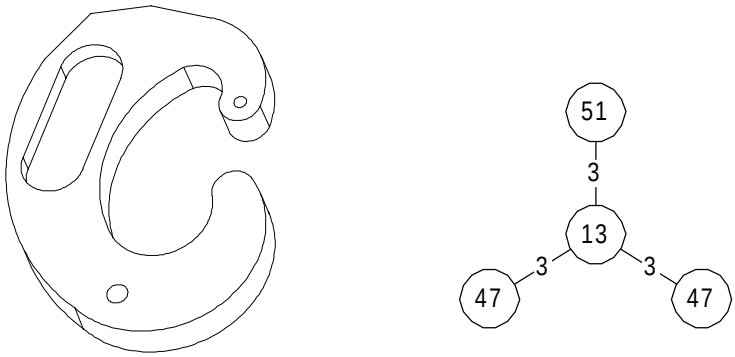


Figure 1.4 : Graphe de géons. Le chiffre à l'intérieur du cercle identifie un géon particulier, alors que la valeur d'arc précise la relation spatiale existant entre deux géons.

Parmi les avantages des descriptions structurelles dans un contexte de vision, on note le fait qu'elles soient invariantes aux transformations affines (translation, rotation, mise à l'échelle), et relativement robustes contre les occlusions et le bruit [16].

Compte tenu des attributs symboliques associés aux géons, il est clair que la description par géons est foncièrement qualitative. Il s'agit d'une différence fondamentale avec les approches par modèles analytiques : la description par géons vise effectivement à désigner la famille à laquelle appartient l'objet, plutôt que l'individu précis. La description par géons est générique, dans la mesure où elle fait totalement abstraction des éléments qui particularisent les diverses instances d'une même famille.

Cet aspect générique des géons restreint d'ailleurs la portée de la théorie de *Reconnaissance par composantes* pour l'identification d'objets. Nous mentionnerons trois problèmes qui limitent le pouvoir discriminant des géons :

1. Un même cylindre généralisé peut éventuellement être représenté par plusieurs géons différents. En témoigne par exemple l'objet de la figure 1.5, dont le profil générateur peut être autant une des petites surfaces carrées qu'une des surfaces horizontales en

forme de U. L'identité du géon dépendra évidemment de l'interprétation qui est faite de cet objet.

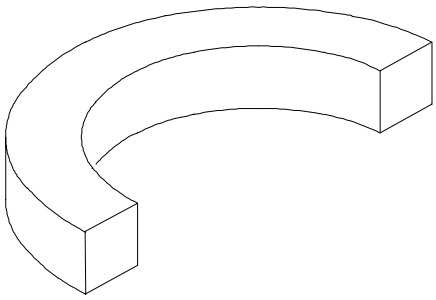


Figure 1.5 : Cylindre généralisé représenté par deux géons différents.

Le parallélépipède est un autre cas du même genre. Ce cylindre généralisé n'est descriptible que par un seul type de géon, dont l'axe de balayage possède cependant trois orientations possibles. Un objet plus complexe dont ferait partie ce géon pourrait alors être décrit de trois façons différentes, selon la façon de codifier la relation spatiale entre géons. Bien qu'il serait possible de préférer une orientation particulière de l'axe de balayage, par exemple celle de l'axe le plus long, cette directive deviendrait inapplicable lorsque le parallélépipède possède au moins deux grands axes de même longueur. L'interprétation d'un paralléllépipède dans une image peut souffrir du même inconvénient, selon l'angle d'observation et le rapport des longueurs d'axes.

2. Le cas précédent soulignait qu'à un cylindre généralisé peut correspondre plus d'un géon ; cette situation s'avère cependant peu fréquente globalement. Par contre, à un géon donné correspond toujours une multitude d'instances de cylindres généralisés. La figure 1.6 illustre par exemple deux objets différents, mais pourtant décrits par les mêmes attributs, donc par le même géon.

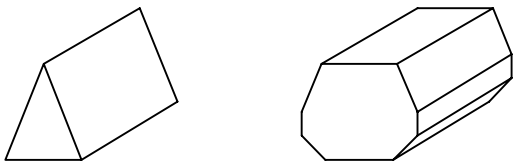


Figure 1.6 : Géon instancié par des cylindres généralisés différents.

3. La représentation par géons ne comporte aucune composante dimensionnelle. Ainsi, des objets morphologiquement similaires mais distincts par un seul facteur d'échelle auraient la même description géon, excluant toute discrimination précise.

Dans le cadre de notre projet, seul le premier de ces inconvénients est susceptible de nous affecter, puisque notre travail consiste à *géoniser* les cylindres généralisés de l'objet. Les deux autres problèmes ne nous affectent pas directement, puisque notre objectif est seulement de produire la description géon d'un objet, nonobstant d'éventuelles collisions ultérieures entre les descriptions. Il appartiendra cependant au système d'inspection automatisée de résoudre ce genre de conflit, l'exclusion de pièces semblables étant incompatible avec la finalité du projet global.

Il est à noter finalement que si l'approche géon est inappropriée pour les applications de précision (métrologie, positionnement etc.), elle pourrait par contre s'avérer particulièrement avenue comme clé de recherche dans une base de données : Biederman calcule en effet que plus de 154 millions d'objets qualitativement différents peuvent être adressés à partir de toutes les combinaisons possibles de trois géons.

1.3 Terminologie de la modélisation solide

Pour les besoins de notre analyse, il sera admis qu'un *objet* est un corps solide d'une seule pièce, constitué d'entités jointes les unes aux autres. Par exemple, deux morceaux distincts, et séparés physiquement, ne pourront être considérés comme faisant

partie d'un seul et même objet. De plus, nous admettrons implicitement qu'un objet est un polyèdre topologiquement valide, c'est-à-dire répondant à l'équation d'Euler-Poincaré :

$$V - E + F - H = 2\,(C - G)$$

avec V : nombre de sommets de l'objet
E : nombre d'arêtes
F : nombre de surfaces
H : nombre de trous dans toutes les surfaces
C : nombre de volumes fermés
G : nombre de trous passant à travers l'objet

Cette équation définit un solide cohérent (*manifold model*), c'est-à-dire physiquement réalisable ; la figure 1.7 illustre le cas de solides qui ne satisfont pas l'équation d'Euler.

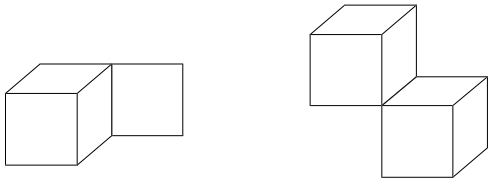


Figure 1.7 : Solides incohérents. L'objet à droite de la figure n'est pas réalisable, puisque l'arête qui réunit les deux cubes est une entité infiniment mince.

Rappelons au passage certaines prémisses géométriques fondamentales des corps solides :

- Une *surface* d'un objet est une portion infiniment mince de l'espace, définie sur domaine continu, et qui constitue l'extrême limite entre l'intérieur et l'extérieur de

l'objet. La terminologie anglophone utilise plutôt le mot *face* pour exprimer l'entité physique, et *surface* pour exprimer l'entité mathématique. Sauf indication contraire, nous référons implicitement à l'entité physique.

- Une *arête* est le lieu d'intersection de deux, et seulement deux, surfaces de l'objet. Noter que la discontinuité de dérivée première n'est pas une définition satisfaisante de l'arête puisque, par exemple, la jonction d'une surface plane avec une surface cylindrique n'engendre pas nécessairement de discontinuité.

- Un *sommet* est le lieu d'intersection d'au moins deux arêtes de l'objet. En pratique, un sommet défini par seulement deux arêtes est plutôt l'exception : l'apex d'un cône est l'un des rares cas du genre parmi les primitives géométriques traditionnelles. En général, un sommet sera le lieu de jonction d'au moins trois arêtes.

Dans un contexte de contrôle de qualité, le terme de *modèle* réfère à l'objet qui sert de référence pour l'inspection, et dont la description géométrique est entièrement connue. La *modélisation* est l'activité qui consiste à définir un objet quelconque à l'aide d'un logiciel de CAO ; de tels logiciels sont plus spécifiquement connus sous le vocable de *modélisateurs* (ou *modeleurs* en France).

Mentionnons finalement que la *géométrie* réfère généralement au type des entités (plan, ligne droite, etc.), que la *morphologie* réfère à leur forme, alors que la *topologie* met l'emphase sur les relations d'adjacence entre ces entités.

1.4 Caractéristiques

La littérature spécialisée dans la reconnaissance volumétrique de données CAO utilise abondamment le terme de *caractéristique* (*feature*), qui décrit « toute constituante physique ou générique d'un objet, et possédant une signification particulière en terme

d'ingénierie » [17]. De façon générale, une caractéristique est un élément qui définit un certain aspect de l'objet, aspect dont les propriétés sont connues et prévisibles, et qui permettrait d'effectuer un raisonnement sur le comportement de cet objet.

Bien qu'il existe plusieurs classifications possibles, nous admettons quatre grandes familles de caractéristiques, à l'instar de [18] :

1. Les caractéristiques géométriques : ce sont les éléments qualitatifs qui décrivent par exemple la forme d'une partie de l'objet (rainure, chanfrein, etc.), ainsi que les paramètres quantitatifs conventionnels, tels les dimensions et les tolérances.

2. Les caractéristiques structurelles : intrinsèques à l'objet, les caractéristiques structurelles servent à définir le solide sous un angle plus global : matériau de fabrication, état de finition des surfaces, etc.

3. Les caractéristiques fonctionnelles : elles désignent les entités de l'objet en terme de leur fonctionnalité respective. Ces caractéristiques viennent en quelque sorte compléter la connaissance de l'objet, en lui ajoutant une dimension initialement absente des caractéristiques purement géométriques. Par exemple, une "cavité cylindrique" deviendra le "logement du roulement à billes" ; un disque situé à l'extrémité d'un corps cylindrique sera possiblement décrit comme "bride de retenue", etc. Bref, les caractéristiques fonctionnelles serviront à préciser l'intention du concepteur, l'usage de destination, le mode d'assemblage, etc.

4. Les caractéristiques technologiques : elles concernent plutôt l'aspect fabrication et analyse de la pièce ; il peut s'agir par exemple des traitements thermiques requis, de la performance attendue sous certaines conditions, etc.

La teneur des caractéristiques dépend fortement du point de vue considéré : par exemple, un ingénieur en fabrication a une perspective toute autre sur un objet que celui qui procède à l'analyse de contrainte. Les caractéristiques extraites d'un objet ont

effectivement des significations complètement différentes, bien qu'elles puissent référer exactement aux mêmes entités géométriques d'un objet.

À date, les caractéristiques de forme (*form features*), cas particulier des caractéristiques géométriques, ont été le plus souvent étudiées puisqu'elles servent souvent à inférer d'autres caractéristiques plus abstraites. La figure 1.8 illustre un ensemble typique de caractéristiques de forme. Les caractéristiques de forme sont l'objet essentiel et unique de notre étude.

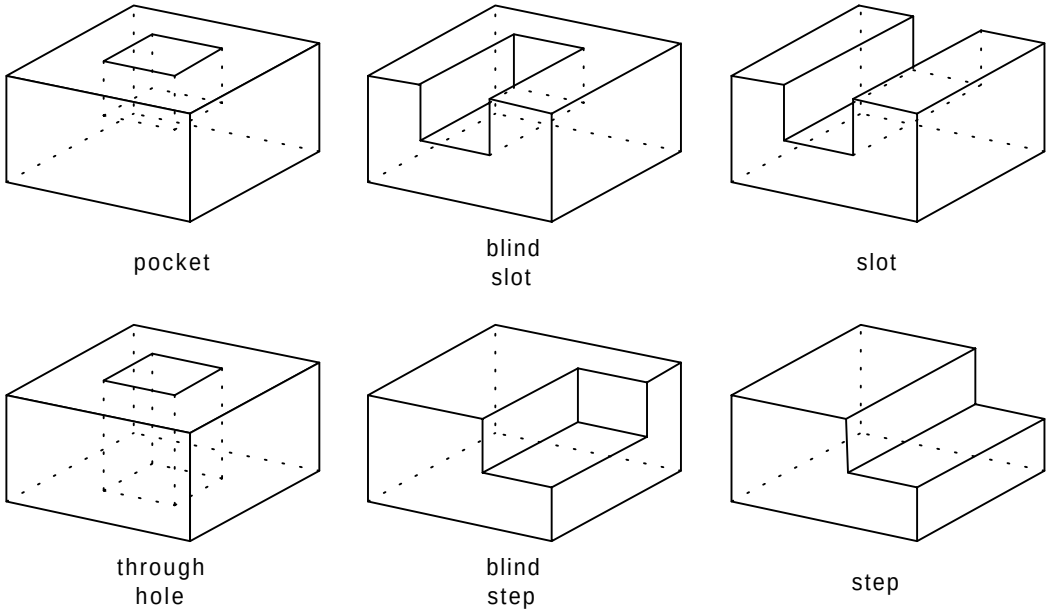


Figure 1.8 : Caractéristiques de forme usuelles. (Adapté de [45])

1.4.1 Caractéristiques de forme STEP

L'émergence des environnements CIM (*Computer Integrated Manufacturing*) a accéléré le besoin d'ajouter de nouvelles couches sémantiques aux descriptions informatiques. L'objectif est de décrire plus globalement un objet en fonction de finalités variées, mais complémentaires : conception, analyse, planification, assemblage,

fabrication, inspection, etc. Le problème de fond concerne l'intégration en un ensemble cohérent de toutes les données du cycle manufacturier, autant celles relatives au produit que celles qui réfèrent aux procédés.

À cet égard, les autorités compétentes élaborent présentement la norme ISO 10303, mieux connue sous le nom de STEP (*STandard for Exchange of Product data*), et rebaptisée PDES (*Product Data Exchange using STEP*) du côté américain. Il s'agit d'un projet d'envergure, qui vise à adapter aux besoins actuels les modes de représentation des données, ainsi que leurs mécanismes d'échange. La norme STEP définirait entre autres un seul format de fichier, applicable à tous les domaines d'activité manufacturière, ce qui faciliterait d'autant le dialogue entre les diverses applications concernées par un même produit. Actuellement, ces applications utilisent un vocabulaire et une sémantique spécifiques à leur fonction, ce qui exclut souvent toute forme de communication entre elles.

Il n'est pas opportun pour nous d'entrer dans le détail de la norme STEP ; dans le cadre de ce projet, nous ne nous intéressons qu'à la partie 48 de cette norme, qui définit justement les caractéristiques de forme génériques d'un corps solide [19]. Au sein de STEP, ces caractéristiques sont classées en trois catégories :

1. Les caractéristiques de volume, qui définissent les volumes élémentaires de l'objet ;
2. Les caractéristiques de transition, qui qualifient la jonction entre les entités ;
3. Les caractéristiques de disposition, qui spécifient la disposition d'un patron répétitif.

La figure 1.9 illustre l'arborescence des caractéristiques de forme STEP, ainsi que leurs instances proposées[1].

---

[1] Il semble en fait que STEP ait abandonné la normalisation des caractéristiques depuis lors.

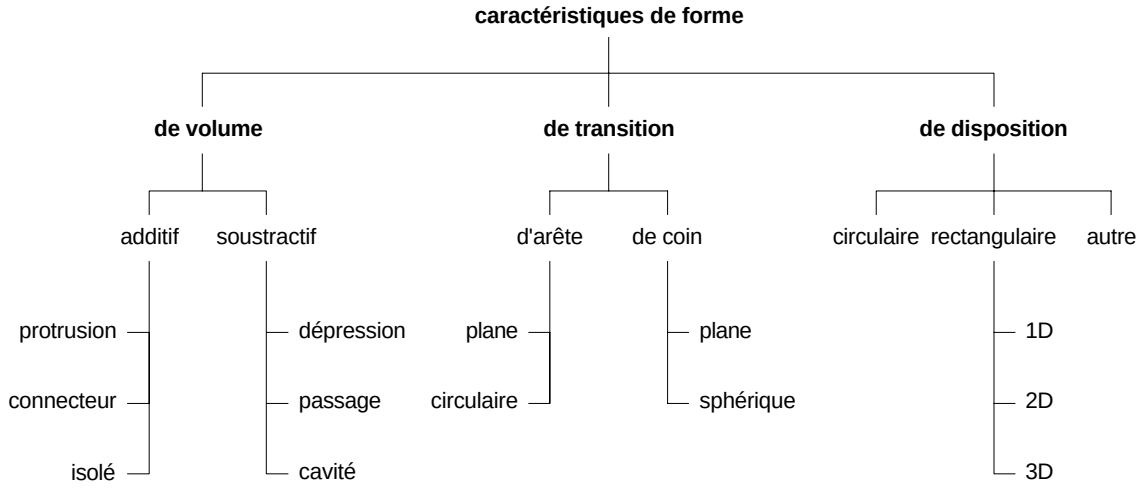


Figure 1.9 : Caractéristiques de forme STEP. (Tiré de [18])

Dans le contexte de ce projet, seules les caractéristiques de volume et de transition intéressent nos travaux. Étant donné l'ampleur du travail à réaliser, nous n'avons d'ailleurs considéré que les caractéristiques de volume, plus fondamentales que les transitions.

On doit surtout retenir que la classification STEP résume la terminologie commune aux différents éléments de forme rencontrés dans la littérature ; selon le cas, les catégories sont plus ou moins détaillées, plus spécifiques au problème étudié, etc. Notre projet ne nous oblige pas vraiment à souscrire à une quelconque norme, puisqu'il est surtout destiné à évoluer dans son univers propre ; notre adhésion à la terminologie STEP vise cependant à éviter une dispersion des concepts entre les différentes méthodologies existantes.

De façon informelle, les six caractéristiques de volumes admises par STEP se définissent comme suit :

- protrusion : volume additif protubérant à l'objet, et qui se termine dans le vide
- dépression : volume soustractif qui crée une seule ouverture sur l'objet
- connecteur : volume additif qui touche la pièce en deux endroits distincts
- passage : volume soustractif qui crée deux ouvertures sur les surfaces de l'objet
- isolé : volume additif qui ne touche aucun des éléments existants de l'objet
- cavité : volume soustractif complètement contenu dans l'objet

Dans le cadre de notre travail, nous admettons implicitement ces définitions, mais excluons d'emblée les deux dernières caractéristiques. D'une part, un volume isolé est inadmissible dans la logique de notre projet, puisqu'il contredit la définition que nous nous sommes donnée d'un "objet" (§ 1.3) ; selon cette définition, un volume isolé ne peut être qu'un objet distinct, et jamais une partie d'un objet existant.

D'autre part, notre rejet des cavités découle d'une perspective pragmatique sur le problème, puisqu'une telle entité n'est pas manufacturable en soi. En pratique, une cavité n'est jamais produite que par l'assemblage de deux dépressions, situées en vis-à-vis sur deux pièces différentes. À l'égard des cavités, nos algorithmes ne sont donc concernés que par la reconnaissance des dépressions. Les différentes caractéristiques de forme admises par ce projet sont illustrées à la figure 1.10.

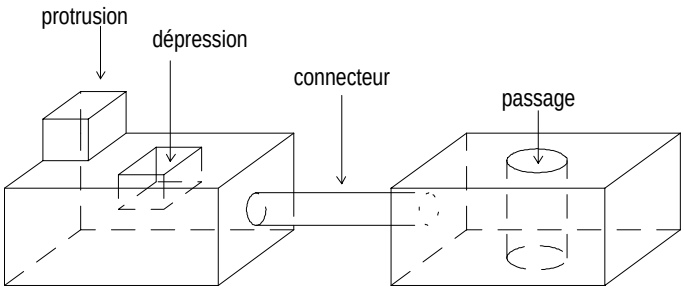


Figure 1.10 : Caractéristiques de forme du projet.

1.5 Forme de l'information CAO

Sachant que notre programme doit extraire les géons de l'objet à partir de sa description CAO, il sera tout à fait pertinent d'étudier au préalable la forme de cette information.

Une description CAO vise à codifier la géométrie de l'objet modélisé ; son objectif est de permettre la mémorisation efficace de l'information dans un fichier, et d'en assurer une interprétation non-ambigüe lors de la relecture. La description CAO vise d'ailleurs cette seule fin, et ne doit pas être confondue avec la représentation interne de l'objet, ou sa visualisation à l'écran. La figure 1.11 résume justement le flux d'information entre ces trois modules.

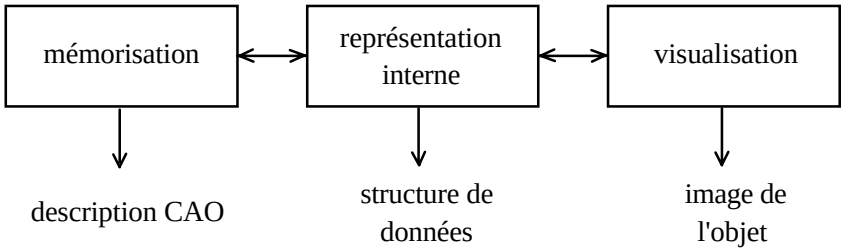


Figure 1.11 : Flux d'information CAO.

La description CAO réfère en pratique à un certain format de fichier, qui sert de source pour la modélisation ou l'analyse des données ; la représentation interne en est la traduction informatique dans une structure de données consistante pour le logiciel. Cette structure de données, même si elle demeure invisible à l'utilisateur, présentera éventuellement un aspect très différent de la forme mémorisée sur disque. La visualisation finalement est l'activité de traduction graphique de l'objet, sous l'angle d'observation choisi par l'utilisateur. La description CAO n'est jamais élaborée à partir de l'image de l'objet : c'est plutôt l'inverse qui est vrai, dans la mesure où la représentation interne de l'objet sert à calculer l'apparence de l'objet à l'écran.

À l'opposé des systèmes de vision artificielle, qui manipulent une information 2D ou 2½D, l'information CAO est véritablement 3D, puisque la description de l'objet est complète. En effet, les représentations symboliques utilisées en vision traduisent une connaissance partielle de l'objet, dépendante de l'angle d'observation ; à l'inverse, une description CAO explicite toutes les entités de l'objet, indépendamment de sa représentation à l'écran.

Une description CAO demeure essentiellement une énumération des entités qui composent l'objet ; cependant, plusieurs facteurs contribuent à lui donner une forme spécifique :

- le type de représentation utilisée, qui réfère à la façon de conceptualiser l'objet. Ces représentations se subdivisent globalement en trois familles, présentées par la suite ;
- le type d'entités admises, dont la teneur sémantique pourra être plus ou moins élevée ;
- le nombre de primitives possibles pour chaque entité ;
- le degré de correspondance entre l'objet et sa description, qui autorisera une modélisation exacte ou approximative du corps ;
- le langage mathématique utilisé pour décrire l'information.

Les nombreux formats de fichier CAO existants traduisent justement cette multiplicité des formes descriptives. On remarquera d'ailleurs que la modélisation proprement dite peut souvent s'accommoder de plusieurs types de descriptions, d'autant plus que l'utilisateur d'un logiciel de CAO reste généralement indifférent à la façon dont les données sont mémorisées. En fait, c'est surtout l'application finale qui dicte le choix d'une description spécifique, chacune étant plus ou moins apte à satisfaire des besoins logiciels précis.

1.5.1 Représentations par décomposition spatiale

Dans les représentations par décomposition spatiale, les objets sont conçus par assemblage d'éléments volumétriques simples, tel qu'illustré à la figure 1.12. Le schéma le plus connu correspond à la description par *voxels*, où l'espace est discrétisé en une série de cubes de dimensions identiques : l'objet est alors décrit par simple énumération des cubes qu'il occupe. Fréquemment utilisé pour la visualisation tridimensionnelle, ce type de représentation demeure cependant peu usité pour la modélisation géométrique, étant donné la faible valeur sémantique associée à chaque entité, et aussi parce qu'il requiert beaucoup d'espace mémoire pour décrire les objets avec une résolution suffisante.

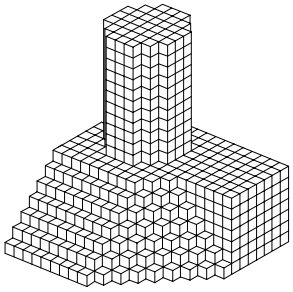


Figure 1.12 : Voxels

L'*arbre octal* (*octree*) est un raffinement de la voxellisation, pour lequel les cubes sont hiérarchisés en éléments de plus en plus petits, selon leur niveau dans l'arbre. Pour fins de simplification visuelle, nous illustrons à la figure 1.13 l'arbre quaternaire (*quad-tree*), l'équivalent 2D de l'arbre octal. Bien qu'elle diminue l'espace mémoire requis, cette représentation est cependant coûteuse pour toute inférence géométrique, puisque les données voisines sont réparties dans différents noeuds de l'arbre.

D'autres schémas admettent un découpage en éléments non-rectangulaires, voire même irréguliers ; les maillages d'analyse par volumes finis en sont un exemple.

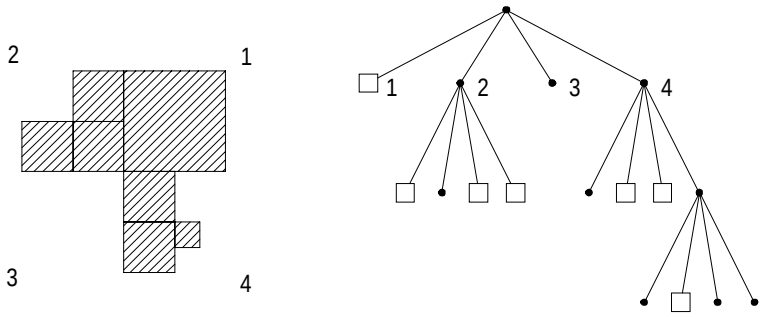


Figure 1.13 : Arbre quaternaire.

1.5.2 Représentations constructives

Les représentations constructives émanent d'une volonté de soumettre le domaine à des lois mathématiques précises, provenant en l'occurrence de la théorie ensembliste. Les objets sont définis à partir de primitives volumétriques paramétrables, sur lesquelles agissent les opérateurs booléens d'union, d'intersection, de complément, et de différence ; l'objet résultant est généralement mémorisé sous forme d'arborescence.

La description CGS (*Construction Géométrique Solide*), illustrée la figure 1.14, est la plus connue de ces méthodes de représentation. Il s'agit d'une description compacte, qui assure la validité de l'objet au sens d'Euler grâce aux propriétés des opérateurs booléens. Cependant, bien que la plupart des modélisateurs commerciaux manipulent les descriptions CGS, que soit pour définir l'objet ou pour représenter l'historique de conception, aucun d'entre eux ne permet à notre connaissance de mémoriser l'arborescence résultante. En pratique, il n'existe d'ailleurs aucun format de fichier standard pour décrire un objet sous forme CGS.

En principe, la description CGS serait particulièrement appropriée pour décrire un objet en fonction de ses composantes volumétriques, tel que nous souhaitons le faire dans ce projet. Cependant, outre la difficulté de générer une telle description avec un

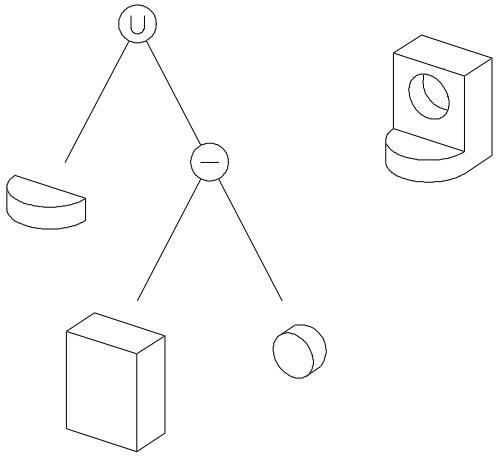


Figure 1.14 : Description CGS.

modélisateur commercial, la représentation CGS possède l'inconvénient majeur d'être non-exclusive pour un objet. Effectivement, si à un arbre CGS donné correspond un et un seul objet, à un objet donné correspond une infinité d'arbres CGS. Ceci traduit l'évidence qu'un même objet peut être construit de plusieurs façons, avec des primitives différentes, tel qu'illustré par exemple à la figure 1.15.

Ainsi que l'ont démontré les travaux antérieurs avec des représentations CGS [20–22], cette multiplicité de solutions complique notablement la tâche des algorithmes de reconnaissance de forme.

Parmi les autres représentations constructives existantes, citons aussi la méthode des demis-espaces (aussi dite BSPT, pour *Binary Space Partition Tree*), qui procède par subdivision progressive de l'espace avec des primitives de surface ; les contraintes imposées à un groupe de primitives permettent alors de définir un volume fermé. Cette approche n'est pas foncièrement différente de la méthode CGS, sinon que les entités disponibles pour construire un objet sont moins évoluées.

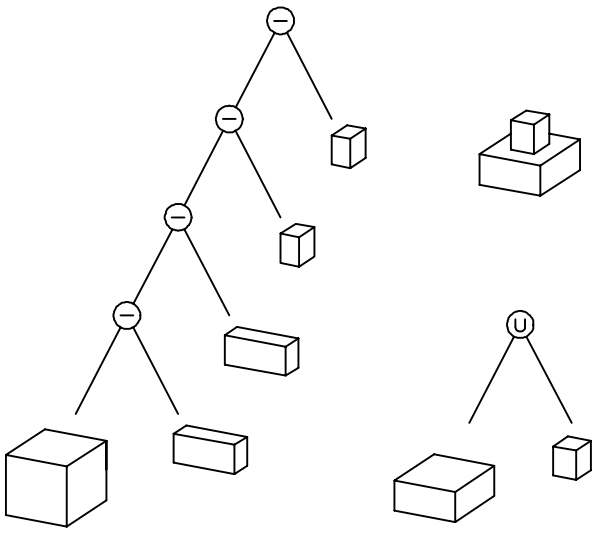


Figure 1.15 : Multiplicité des solutions CGS.

1.5.3 Représentation par les limites

Les représentations par les limites, communément dites *Brep* (pour *Boundary Representation*) décrivent uniquement l'enveloppe extérieure du corps par énumération de ses surfaces, en qualifiant autant leur géométrie (type de surface, dimensions, etc.) que leur topologie, c'est-à-dire leur relation d'adjacence. La figure 1.16 illustre ce schéma descriptif, qui définit l'objet comme un assemblage de surfaces, spécifiées en fonction des arêtes qui délimitent leur extension spatiale.

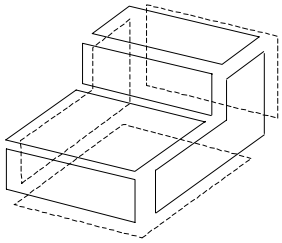


Figure 1.16 : Représentation Brep. Les surfaces de l'objet sont explicitées une à une, ainsi que les entités inférieures qui les composent (arêtes, sommets).

Ce type de représentation correspond d'ailleurs au schéma de modélisation le plus répandu en industrie. Nous élaborerons un peu plus sur cette représentation, puisqu'elle correspond au format des données d'entrée de notre projet.

a) Formes énumératives

Trois familles d'entités sont susceptibles d'apparaître dans une représentation par les limites, à savoir les surfaces, les arêtes et les sommets, notés respectivement F, E et V[2]. S'il est nécessaire de spécifier toutes les surfaces, arêtes et sommets d'un objet donné, il existe cependant plusieurs approches énumératives ; nous citons les plus communes d'entre elles, illustrées à la figure 1.17 :

- F (E) : dans ce modèle, toutes les surfaces sont définies en fonction des arêtes qui les bordent ; les arêtes sont énumérées séparément, et les coordonnées de leurs extrémités sont spécifiées pour chaque instance d'arête.

- F (V) : les surfaces sont définies selon les sommets qui marquent l'extrémité de leurs arêtes, et les coordonnées de chaque sommet sont énumérées dans une deuxième partie de la description.

- F (E (V)) : les surfaces sont définies par leurs arêtes, énumérées par la suite. Ces arêtes sont spécifiées en fonction de leurs sommets, qui font l'objet d'un listage séparé.

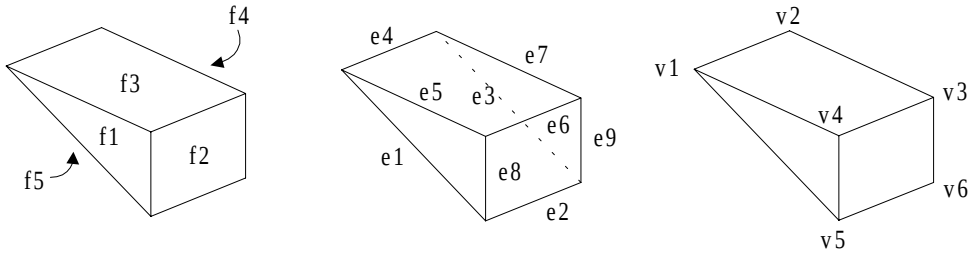
Ces formes énumératives ont chacune leurs particularités : la forme F(E) par exemple souffre d'une évidente redondance d'information. La duplication des coordonnées des sommets devient d'ailleurs coûteuse en reconnaissance de forme, puisqu'une modification sur un sommet doit alors être propagée sur au moins deux arêtes supplé-

---

2 Nous utilisons les symboles usuels de la littérature anglophone, soit F pour *face*, E pour *edge* et V pour *vertex*.

mentaires. Les entités doivent aussi être manipulées avec prudence en raison des erreurs d'arrondi, communes en modélisation.

La description F(V) offre l'avantage sur la précédente de faciliter les manipulations de sommets, puisque leurs coordonnées (x,y,z) respectives sont regrou-



F (E)

```
f1 = e1 e8 e5
f2 = e2 e9 e6 e8
f3 = e6 e7 e4 e5
f4 = e9 e3 e7
f5 = e2 e1 e4 e3
```

⟹

```
e1 = x1 y1 z1 x5 y5 z5
e2 = x5 y5 z5 x6 y6 z6
e3 = x6 y6 z6 x2 y2 z2
e4 = x2 y2 z2 x1 y1 z1
e5 = x1 y1 z1 x4 y4 z4
e6 = x4 y4 z4 x3 y3 z3
e7 = x3 y3 z3 x2 y2 z2
e8 = x4 y4 z4 x5 y5 z5
e9 = x3 y3 z3 x6 y6 z6
```

F (V)

```
f1 = v1 v5 v4
f2 = v5 v6 v3 v4
f3 = v4 v3 v2 v1
f4 = v6 v2 v3
f5 = v6 v5 v1 v2
```

⟹

```
v1 = x1 y1 z1
v2 = x2 y2 z2
v3 = x3 y3 z3
v4 = x4 y4 z4
v5 = x5 y5 z5
v6 = x6 y6 z6
```

F (E (V))

```
f1 = e1 e8 e5
f2 = e2 e9 e6 e8
f3 = e6 e7 e4 e5
f4 = e9 e3 e7
f5 = e2 e1 e4 e3
```

⟹

```
e1 = v1 v5
e2 = v5 v6
e3 = v6 v2
e4 = v2 v1
e5 = v1 v4
e6 = v4 v3
e7 = v3 v2
e8 = v4 v5
e9 = v3 v6
```

⟹

```
v1 = x1 y1 z1
v2 = x2 y2 z2
v3 = x3 y3 z3
v4 = x4 y4 z4
v5 = x5 y5 z5
v6 = x6 y6 z6
```

Figure 1.17 : Formes énumératives en Brep

pées en une seule liste. Cette forme restreint cependant le domaine de modélisation, faute d'une énumération explicite des arêtes. En effet, la forme F(V) associe implicitement une arête à chaque paire de sommets ; à défaut d'être caractérisées explicitement, comme en F(E), ces arêtes sont donc par convention des lignes droites. Cette dernière contrainte implique alors que les objets décrits en F(V) soient polyédriques, excluant ainsi les corps de forme arrondie. Il demeure possible par contre d'exprimer un objet arrondi en F(V), en autant qu'on ne requière pas une description exacte ; les surfaces arrondies seront alors approximées par un ensemble de surfaces planes, dites *micro-facettes*.

La forme F(E(V)) est selon nous la méthode descriptive la plus efficace pour l'analyse, autant pour l'exhaustivité de l'information que pour sa robustesse aux erreurs de calculs. La double indirection entraîne par contre un plus grand parcours de recherche dans la description pour accéder à l'information désirée.

On doit noter au passage que les formes énumératives illustrées à la figure 1.17 sont réduites ici à leur plus simple expression ; en pratique, il est courant que de nombreux autres descripteurs soient associés à chaque entité afin d'accroître le vocabulaire descriptif. Par exemple, une surface caractérisée uniquement par ses arêtes doit être un plan ou une surface gauche ; en ajoutant d'autres éléments descriptifs, il devient ainsi possible de décrire des surfaces plus complexes.

b) Boucles

Dans la terminologie de la modélisation, la liste des arêtes qui bordent une surface est dite *boucle*. Tel que son nom l'indique, une boucle est une séquence ordonnée d'arêtes qui forment un parcours fermé. De nombreux formats de fichiers CAO prévoient effectivement une rubrique "boucle" parmi les descripteurs de surfaces.

La pertinence d'énumérer les arêtes qui délimitent une surface est évidente. Par exemple, si on décrit une surface plane d'un objet par énumération des paramètres scalaires A, B, C et D, de l'équation algébrique Ax + By + Cz + D = 0, on définit en fait un plan infini, coplanaire avec la surface de l'objet ; ceci contreviendrait aux exigences d'une description Brep, qui n'autorise que des entités de dimensions finies, localisées à un endroit unique de l'espace. L'énumération des arêtes limitrophes devient alors une méthode simple pour délimiter l'extension spatiale de la surface concernée.

Pour certains objets, il est utile d'admettre l'existence de boucles externes et de boucles internes : les unes définissent le contour extérieur d'une surface, tandis que les autres circonscrivent les endroits où la surface n'est pas définie, à l'intérieur de la boucle externe. La figure 1.18 illustre bien la difficulté à résoudre : le plan F1 est défini partout à l'intérieur de la boucle L1, sauf dans la cavité carrée, et sous le cylindre. Il devient donc nécessaire de qualifier F1 par les trois boucles L1, L2 et L3 pour obtenir une description complète de cette surface.

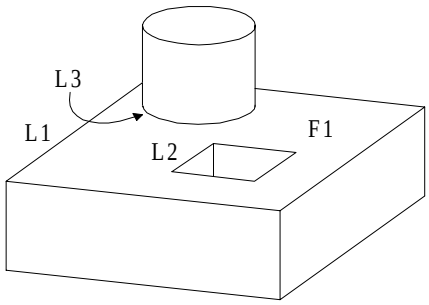


Figure 1.18 : Boucles internes et externes

À remarquer que si les boucles internes sont requises pour mieux circonscrire le domaine de définition d'une surface, elles délimitent aussi le lieu de jonction entre la surface et certaines des dépressions et protrusions qui s'y rattachent ; cette propriété nous sera utile pour l'extraction des géons. Nous ajouterons aussi les postulats suivants, qui découlent de l'évidence géométrique :

- Postulat 1 : Une surface possède nécessairement une et une seule boucle externe.
- Postulat 2 : Une surface possède zéro, une ou plusieurs boucles internes.

Le sens d'énumération des arêtes au sein d'une boucle n'est jamais arbitraire puisqu'il indique le côté matériel de la surface. S'il est évident que les arêtes d'une boucle seront énumérées en ordre séquentiel, quelque soit le point de départ, reste cependant que l'énumération peut se faire dans le sens horaire ou antihoraire (souvent dit *sens trigonométrique*). En pratique, le sens d'énumération suit une convention propre au format de fichier. Une règle de la main droite par exemple implique que les arêtes qui bordent une surface soient énumérées dans le sens antihoraire, avec le pouce qui pointe vers l'extérieur de l'objet, donc dans la même direction que la normale à la surface ; les boucles mentionnées à la figure 1.17 respectent cette convention.

Il est d'ailleurs fréquent que la normale soit explicitement décrite parmi les attributs d'une surface. Cependant, il devient impossible d'attribuer une normale unique à une surface non-plane : à cet égard, le sens d'énumération est donc un indicateur plus universel pour calculer la position du matériau par rapport à la surface.

Si certains formats de fichier présument une convention de main gauche, d'autres encore admettent indifféremment les deux conventions, mais préciseront à l'aide d'un commutateur laquelle des deux est en vigueur pour une description Brep donnée ; il est cependant impératif que la même règle s'applique pour toutes les surfaces du modèle.

Remarquons finalement que le sens de parcours des arêtes d'une boucle interne est toujours à l'inverse du sens de parcours des boucles externes. L'uniformité du sens d'énumération implique en effet qu'une arête soit parcourue dans un sens pour une surface, et dans l'autre sens pour la surface adjacente ; cette règle serait cependant enfreinte si les boucles internes et externes auraient le même sens d'énumération.

La notion de boucle oblige en pratique à découper artificiellement des entités qui devraient apparemment demeurer d'une seule pièce ; il s'agit en l'occurrence d'éviter les cas d'exceptions, toujours problématiques lors du traitement des données.

La figure 1.19 illustre le cas simple d'une surface cylindrique : en (a), le cylindre apparaît tel qu'il aurait été défini par l'usager, soit avec une arête circulaire E1 extrudée le long d'un axe droit, produisant en bout de course une deuxième arête circulaire E2. Il devient alors impossible de satisfaire le postulat 1 avec les arêtes en question, puisqu'il faudrait deux boucles externes L1 = {E1} et L2 = {E2} pour définir la surface cylindrique. Ceci serait d'ailleurs contradictoire avec le concept même de boucle, qui prévoit qu'une surface donnée puisse être complètement circonscrite par un trajet fermé d'arêtes.

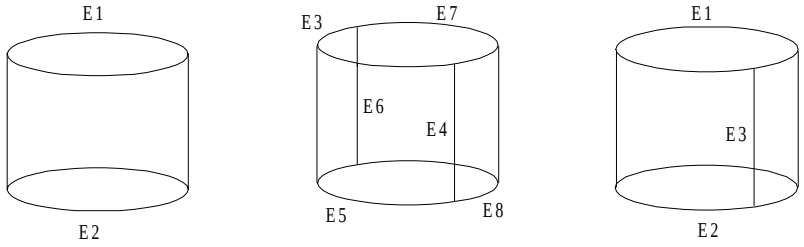


Figure 1.19 : Découpage de boucle. (a) Cylindre original, défini par l'usager. (b) Découpage des surfaces pour satisfaire le postulat 1. (c) Découpage incohérent, à moins d'ajouter une deuxième arête confondue avec E3.

La solution apparaît évidemment en 1.19 (b) : le modélisateur a découpé chaque arête circulaire en deux demis-arcs de cercle, et généré des arêtes droites entre elles ; reste alors deux surfaces en demi-cylindre, dont les boucles externes sont L1 = {E3, E4, E5, E6} et L2 = {E7, E6, E8, E4}. Il est à noter que la solution qui consiste à ajouter une seule arête verticale, tel qu'illustré en 1.19 (c), serait incohérente : la boucle L1 = {E1, E3, E2, E3} ne permettrait pas de retrouver le côté matériel de la surface, quelle que soit la convention en vigueur. Les modélisateurs qui produisent ce genre de solution

doivent nécessairement générer à la place de E3 deux arêtes distinctes, bien que géométriquement confondues.

c) Langages mathématiques

Parmi les éléments qui confèrent une spécificité particulière à une description Brep donnée, il faut citer le langage mathématique utilisé pour décrire les entités ; ce langage sera conditionné par le type de primitives admises pour représenter chaque entité. En effet, un langage simple suffit à représenter des plans, à supposer qu'il s'agisse des seules entités de surface acceptées, alors qu'un langage autrement plus complexe est nécessaire pour décrire des surfaces sculptées. En fait, un sommet d'un objet est le seul type d'entité qui admette une et une seule primitive, à savoir lui-même, et qui n'est jamais représenté que par ses coordonnées (x,y,z), quel que soit le langage mathématique utilisé pour décrire les entités de plus haut niveau.

Globalement, on peut diviser les langages mathématiques en deux familles, soit les langages algébriques et les langages paramétriques. Pour les premiers, les entités sont décrites de façon conventionnelle, c'est-à-dire par une fonction $f(x,y,z) = 0$. Ce type de langage comporte cependant certains inconvénients :

- Les entités admissibles, qu'il s'agisse d'arêtes ou de surfaces, doivent être descriptibles sous une forme algébrique ; les formats Brep qui utilisent un langage exclusivement algébrique doivent donc limiter leur répertoire de primitives en conséquence. Il faut noter par ailleurs que l'exigence de descriptibilité algébrique peut devenir assez contraignante : si par exemple deux surfaces cylindriques peuvent être représentées par des équations algébriques, il devient difficile de décrire algébriquement n'importe quelle intersection de ces deux cylindres (qui engendrent dans le pire des cas une courbe du 4e degré).

- Certaines entités ne peuvent pas être mises sous la forme explicite z = f(x,y), mais doivent demeurer sous la forme implicite f(x,y,z) = 0. Ceci complique l'analyse des données, lorsqu'il faut par exemple déterminer le lieu d'intersection de deux surfaces : le calcul numérique est alors requis, ce qui accroît sensiblement la lourdeur des algorithmes.

- Les manipulations d'entités deviennent notoirement compliquées du point de vue algébrique, surtout lorsqu'elles impliquent des rotations dans l'espace.

- Le risque d'erreur d'arrondi augmente avec les manipulations. En effet, une solution obtenue d'une première équation peut facilement ne pas satisfaire une deuxième équation à cause de problèmes de convergence. L'exemple typique est celui des fonctions trigonométriques, requises pour manipuler des entités courbes, ou pour effectuer une rotation des entités.

Les langages paramétriques cherchent à atténuer ces difficultés en exprimant chaque entité en fonction d'un paramètre, souvent normalisé entre 0 et 1. Une arête **E**, ou une surface **F**, seront alors exprimées sous forme vectorielle :

$$\mathbf{E} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$$

Outre que les représentations vectorielle et matricielle soient particulièrement bien adaptées aux méthodes de calcul par ordinateur, le langage paramétrique présente de nombreux avantages par rapport aux descriptions strictement algébriques :

- Il n'y a aucune restriction quant au type d'entité descriptible paramétriquement. En effet, toutes les primitives géométriques usuelles (droite, arc de cercle, plan, cylindre, etc.) peuvent être paramétrées à l'aide de transformations simples, déjà

tabulées[3]. Les entités plus complexes qui n'ont pas d'expression algébrique concise, tels les splines et les NURBS (*Non-Uniform Rational B-Splines*), sont aussi paramétrables par un découpage en sous-éléments.

- Le langage paramétrique assure une constance de forme de la description. Pour les langages algébriques, le domaine de modélisation est généralement restreint aux entités qui satisfont l'équation générale des quadriques :

$$F(x,y,z) = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz + Gx + Hy + Iz + J = 0$$

  En effet, sans cette contrainte, une primitive quelconque pourrait très bien être non-polynomiale, ce qui produirait rapidement une variabilité descriptive impossible à gérer pour l'ordinateur. Le langage paramétrique ne souffre pas de cet inconvénient, dans la mesure où, au pis-aller, il suffit d'énumérer les points de contrôle de l'entité.

- Les entités sont toujours décrites explicitement plutôt qu'implicitement, ce qui simplifie la résolution symbolique d'équations. De plus, la résolution de variables portera plus souvent sur la valeur du ou des paramètres : ceci contribue à diminuer les erreurs d'arrondi, puisque les coordonnées spatiales d'un point sont toutes calculées à partir de la même valeur du paramètre.

- Les transformations affines (translation, rotation et mise à l'échelle) sont notablement plus simples à effectuer qu'avec le langage algébrique. En effet, en introduisant une coordonnée *w*, on redéfinit l'univers géométrique (x,y,z) en un système (*x*,*y*,*z*,*w*) dit *homogène*, pour lequel :

$$x(t) = x(t) / w$$
$$y(t) = y(t) / w$$
$$z(t) = z(t) / w$$

---

[3] Nous référons notre lecteur à l'annexe B pour des exemples de paramétrisation d'arêtes ou de surfaces.

En pratique, *w* est toujours posé égal à 1 pour assurer l'identité entre l'univers homogène et l'univers non-homogène. La coordonnée *w* n'est qu'un artifice mathématique, sans signification physique ; cette astuce permet cependant d'effectuer n'importe quelle transformation affine par produit matriciel avec une matrice de 4×4 ; par exemple, une entité entièrement translatée selon un déplacement (t_x, t_y, t_z) devient :

$$\mathbf{X}_{\text{translaté}} = \mathbf{X} \times \mathbf{T} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ w \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} = \begin{bmatrix} x(t) + t_x w \\ y(t) + t_y w \\ z(t) + t_z w \\ w \end{bmatrix}$$

Il existe similairement des matrices de rotation et de mise à l'échelle. Une séquence de transformations affines $\mathbf{T}_1$ et $\mathbf{T}_2$ peut d'ailleurs être mémorisée sous forme d'une matrice de transformation unique $\mathbf{T} = \mathbf{T}_1\mathbf{T}_2$, et ceci indépendamment du type d'entité à manipuler.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 Revue bibliographique

Il existe déjà une abondance de publications concernant la reconnaissance d'entités volumétriques à partir d'une description CAO. Ce type de problème est étudié depuis presque vingt ans, surtout dans un objectif de planification de procédés manufacturiers.

Parmi les revues qui font le plus souvent état de progrès dans ce domaine, mentionnons :

- Computer-Aided Design (Elsevier)
- International Journal of Advanced Manufacturing Technology (Springer-Verlag)
- International Journal of Computer Integrated Manufacturing (Taylor & Francis)
- Revue internationale de CFAO et d'informatique graphique (Hermès)
- IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE Society)

Outre les actes de conférence, nous relevons aussi au moins une monographie très directement liée au sujet, celle de Shah et Mäntylä [18].

2.2 Travaux en vision artificielle

À date, l'approche géon a été expérimentée exclusivement dans un contexte de vision artificielle ; ceci n'a rien de très surprenant, puisque les géons émanent de travaux sur la vision humaine. Les travaux effectués en vision artificielle peuvent se diviser sommairement en deux groupes, selon le type de données reçues en entrée.

a) Systèmes opérant à partir de données 2D

Ces systèmes reçoivent en entrée une image d'illuminance ou un dessin au trait (*line drawing*). L'un et l'autre ne sont pas foncièrement différents, dans la mesure où les images d'illuminance préludent invariablement à l'extraction des contours ou des régions de l'image.

Le recouvrement des géons à partir des entités intermédiaires s'effectue selon différentes techniques : Bergevin et Levine [8–9] par exemple exploitent les propriétés spécifiques des arêtes (type de jonctions, parallélisme, colinéarité, etc), tout comme Hummel et Biederman [11] dans leur implémentation neuronique. Dickinson *et al.* [5] procèdent par appariement de primitives, le système ayant mémorisé au préalable les vues caractéristiques des géons. Parmi d'autres méthodes, on pourrait aussi évoquer l'approche par contours actifs de Pilu et Fisher [12], illustrée à la figure 2.1.

b) Systèmes opérant à partir de données 2½D

Moins nombreux que les systèmes 2D, les travaux de cette catégorie utilisent en entrée une image de profondeur (*depth map*), obtenue par télémétrie ou par stéréoscopie. Comme dans le cas des approches 2D, le recouvrement des géons requiert l'extraction initiale des primitives géométriques (arêtes et surfaces) de l'image, selon les techniques de segmentation appropriées. Parmi les méthodes de reconnaissance, nous mentionne-

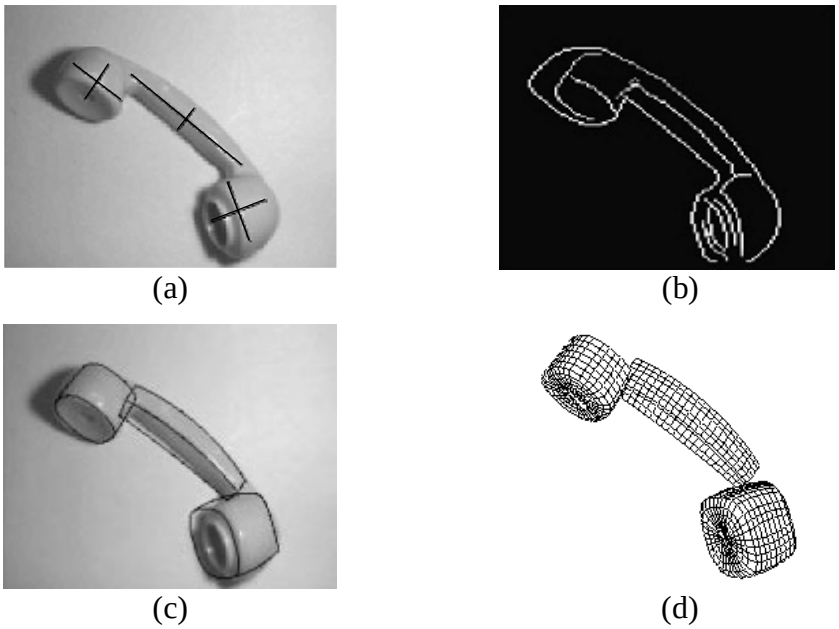


Figure 2.1 : Recouvrement de géons par contours actifs. (a) L'utilisateur indique la position et la dimension approximative des géons. (b) Extraction des contours de l'image, sur lesquels les géons seront ajustés. (c) Fin du processus de relaxation des contours actifs. (d) Reconstruction tridimensionnelle de l'objet. (Tiré de [12])

rons l'ajustement de superquadriques aux données (Raja et Jain [7], Wu et Levine [13]), l'appariement par isomorphisme de graphe (Madsen *et al.* [14]), ainsi que les techniques reposant sur la convexité des arêtes ou la topologie des entités (Nguyen et Levine [15]).

* * *

Bien que le concept de géon soit exclusif à la vision artificielle, nous retenons finalement très peu des travaux dans ce domaine. En effet, les techniques d'analyse de données CAO diffèrent sensiblement des algorithmes d'interprétation d'image pour plusieurs raisons :

- L'image est constituée de pixels, c'est-à-dire d'entités élémentaires ayant peu de rapport direct avec le contenu de la scène ; le travail d'analyse repose en grande

partie sur la conversion et l'agglomération de cette information en entités plus signifiantes. En contrepartie, une description CAO indique dès le départ tous les évênements significatifs de la scène (sommets, arêtes et surfaces) sous une forme mathématique idéale.

- Une part importante du travail de vision consiste à surmonter un problème mal posé, c'est-à-dire à résoudre la multiplicité de solutions qu'entraîne la projection d'une scène tridimensionnelle sur un domaine 2D. Ce problème est inexistant avec les données CAO, puisque l'information est spécifiée et manipulée directement dans le domaine 3D.

- Une description CAO est presque parfaite, dans la mesure où le bruit d'entrée se résume aux erreurs d'arrondi ; à l'inverse, une image réelle contiendra plusieurs éléments indésirables : ombrages, reflets, imperfections de filtrage, etc. Les algorithmes de vision doivent donc identifier et rejeter les entités parasites, inexistantes dans une description CAO.

- L'image traduit une connaissance incomplète de la scène, dans la mesure où les géons doivent être inférés à partir des seuls éléments visibles, et leur parties manquantes déduites en fonction d'hypothèses, de probabilités, etc. Si l'analyse de données CAO n'est pas totalement exempte de ce genre de démarche, elle ne souffre cependant pas d'une connaissance partielle de la scène, puisque l'objet est connu au complet, sous tous ses angles.

Tel que nous le verrons au prochain paragraphe, l'objectif de notre projet s'apparente nettement plus aux travaux d'extraction de caractéristiques effectués en génie mécanique, même si la notion de géon y est pratiquement inconnue.

2.3 Travaux en génie mécanique

Les travaux dans ce domaine s'inscrivent toujours dans un contexte manufacturier. Leur objectif général vise l'automatisation de certaines tâches complexes, telles l'analyse de manufacturabilité, la classification codée (*group technology coding*), ou la planification de la séquence d'opérations requise pour fabriquer une pièce (*CAPP*, pour *Computer Assisted Process Planning*).

Depuis la thèse de doctorat de Kyprianou en 1980 [23], la première qui soit recensée en matière de reconnaissance de forme dans une description CAO, plus d'une trentaine d'articles ont été publiés. A défaut d'une étude détaillée des travaux, nous essaierons plutôt d'identifier les principales tendances, et d'en dégager les idées maîtresses. Pour un état de l'art plus complet, nous référons notre lecteur à l'une ou l'autre des synthèses déjà publiées [24–29].

Notre lecteur voudra bien noter par ailleurs que cette revue de la littérature est nettement plus exhaustive que nécessaire aux seules fins de ce projet. Notre objectif n'est pas tant de déterminer la méthode la plus appropriée pour l'extraction des géons que de rendre compte des méthodes existantes. Seul le paragraphe 2.3.1.2 est directement relié à la méthodologie retenue pour ce projet.

De façon générale, on peut classifier les travaux antérieurs selon un certain nombre d'approches, analysées succintement dans les paragraphes qui suivent. On doit noter que certains travaux recourent à plusieurs techniques à la fois, alors que d'autres utilisent des techniques complètement différentes ; nous avons choisi d'ignorer ces derniers, plus marginaux par rapport aux tendances générales.

2.3.1 Approches par graphe

Les approches par graphe ont été initiées par Ansaldi [30], et ont pris leur essor avec ce que nous appellons "l'école italienne" (de Floriani [31], Falcidieno et Giannini [32]). Cette méthode d'analyse est rapidement devenue un axe de recherche important dans le domaine.

Pour l'essentiel, l'approche par graphe implique la conversion initiale du modèle en un graphe relationnel, où les noeuds représentent les surfaces de l'objet, et pour lequel les arcs expriment les relations d'adjacence entre les surfaces. Il existe aussi de nombreuses variantes à ce schéma, par exemple celui des *hypergraphes de surfaces* (Lentz et Sowerby [33]), où les arcs expriment le voisinage simultané de plusieurs surfaces (plutôt que deux à deux), ou encore les *graphes d'arêtes et sommets* (Chuang et Henderson [34]), dans lesquels l'inférence s'effectue à partir d'entités de plus bas niveau.

Différentes heuristiques de reconnaissance sont mises en oeuvre dans les approches par graphes, fondées principalement sur la topologie des entités. Nous mentionnerons deux stratégies communes en matière de graphes.

2.3.1.1 L'appariement par isomorphisme

Cette technique bien connue en reconnaissance de forme consiste à rechercher dans le graphe un patron pré-déterminé, et dont la séquence correspond à une caractéristique possible de l'objet (Joshi et Chang [35], Sakurai et Gossard [36], Laakko et Mäntylä [37]). Lorsque le patron recherché est identifié, il est ensuite isolé et enlevé du graphe, qui doit finalement être completé tel que le serait l'objet après extraction de la caractéristique. La figure 2.2 illustre l'identification d'une rainure sur un objet simple.

Parmi les avantages et inconvénients de l'isomorphisme, mentionnons :

- Il n'est efficace que pour des objets de morphologie relativement simple. Il suffirait par exemple de quelques modifications à la pièce de la figure 2.2 pour accroître sensiblement la lourdeur des algorithmes de complétion du modèle. Cette dernière remarque ne discrédite cependant pas l'isomorphisme comme technique de localisation de caractéristique.

- Il n'y a pas d'exigence de polyédrisme à imposer aux objets, comme c'est souvent le cas pour d'autres techniques, sinon que pour simplifier le calcul en général.

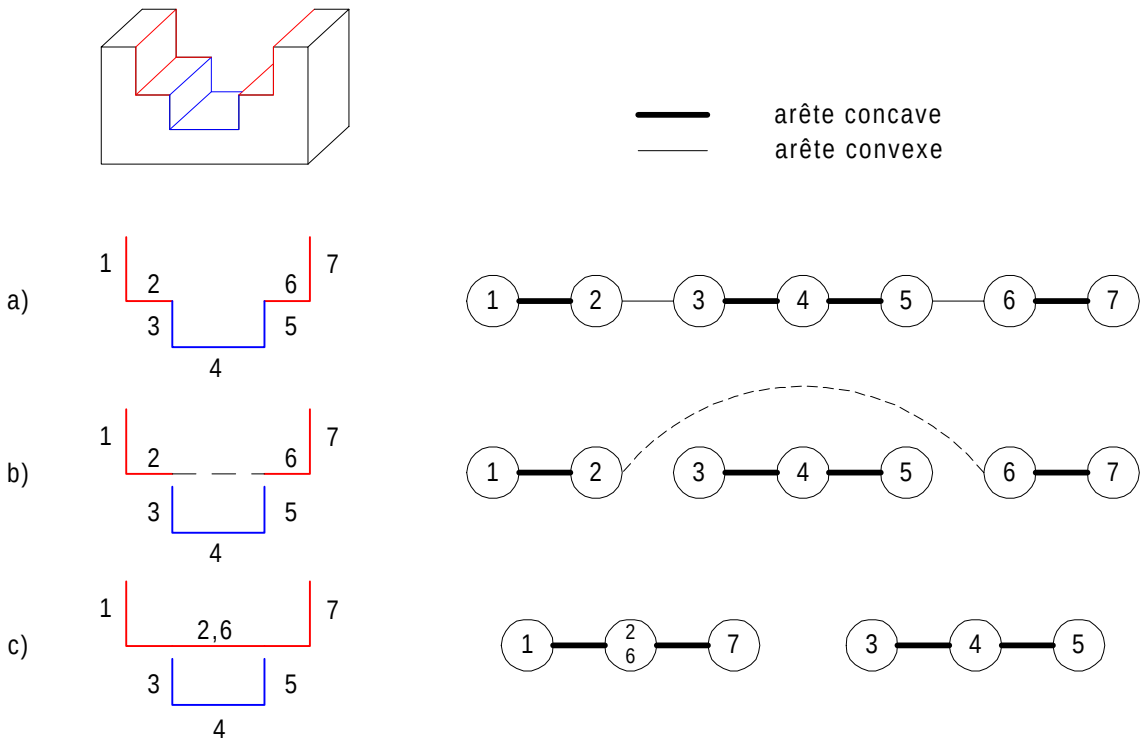


Figure 2.2 : Isomorphisme de graphe. (a) Génération du graphe relationnel (partiellement illustré). (b) Extraction du sous-graphe correspondant à une rainure. (c) Complétion du modèle. Une nouvelle rainure est mise en évidence par cette opération. (Tiré de [37])

- L'isomorphisme devient rapidement prohibitif en temps de calcul, puisque le coût de l'appariement croît exponentiellement avec le nombre de surfaces ; cette technique serait peu appropriée dans un contexte de temps réel avec des objets comportant quelques centaines de surfaces. En pratique, différentes heuristiques doivent être mises en oeuvre pour délimiter un espace de recherche particulier dans le graphe de l'objet.

- Problème critique, le fait que des caractéristiques géométriquement différentes peuvent partager la même topologie, et donc présenter le même graphe (Gardan et Minich [26]). Dans une telle éventualité, le programme parviendrait rapidement à des conclusions erronées, à moins de procéder à une analyse détaillée des noeuds environnants au sous-graphe. Cette démarche augmente la complexité des algorithmes, dans la mesure où le patron recherché ne synthétise plus à lui seul toute l'information requise pour la reconnaissance. La figure 2.3 illustre un cas de ce genre.

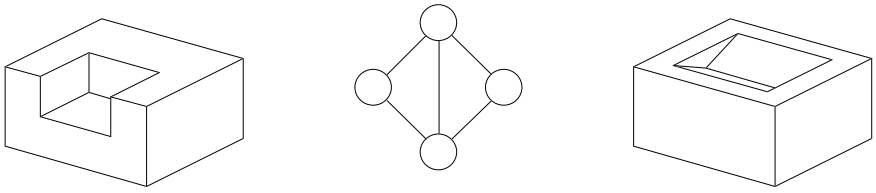
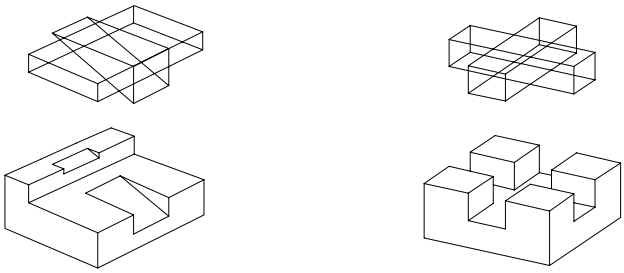


Figure 2.3 : Gemellité des graphes de surfaces. L'encoche et la dépression présentent le même graphe générique, illustré au centre.

- Les *interactions* ont un effet destructeur sur les graphes, dans la mesure où elles masquent rapidement les entités recherchées. Tel qu'illustré à la figure 2.4, les interactions sont le produit de caractéristiques imbriquées l'une dans l'autre. Le recouvrement de caractéristiques en présence d'interactions constitue d'ailleurs l'une des pierres d'achoppement du domaine, quelle que soit la méthode utilisée. Les travaux de Marefat et Kashyap [38] s'intéressent au problème des interactions dans le cadre de l'isomorphisme.



(a) (b)

Figure 2.4 : Interaction de volumes. (a) La marche demeure facilement détectable, mais le programme identifiera probablement deux prismes au lieu d'un seul. (b) Les rainures deviennent très difficiles à reconnaître dans le graphe de l'objet.

2.3.1.2 La méthode des noeuds de séparation

La méthode des noeuds de séparation est inspirée de la théorie des graphes, et consiste à rechercher les noeuds qui permettent de séparer le graphe original de l'objet en plusieurs sous-graphes distincts (Gavankar et Henderson [39]). La figure 2.5 illustre un cas de graphe simplement connecté ; on peut similairement développer des algorithmes de recherche de sous-graphes doublement connectés, voire k-connectés.

Parmi les avantages et inconvénients de cette méthode, mentionnons :

- La technique des noeuds de séparation ne permet pas d'identifier une caractéristique, mais seulement de la localiser dans le graphe de l'objet. Reste que l'identification ultérieure s'en trouve en principe simplifiée, dans la mesure où l'algorithme doit manipuler moins d'entités.

- Les caractéristiques extraites du graphe sont génériques, c'est-à-dire qu'elles sont définies en fonction de critères strictement topologiques, et indépendants de la forme spécifique du volume. Ceci s'oppose nettement à l'approche isomorphique, où le programme recherche uniquement des entités prédéfinies. L'algorithme permet aussi de manipuler des descriptions Brep non-polyédriques.

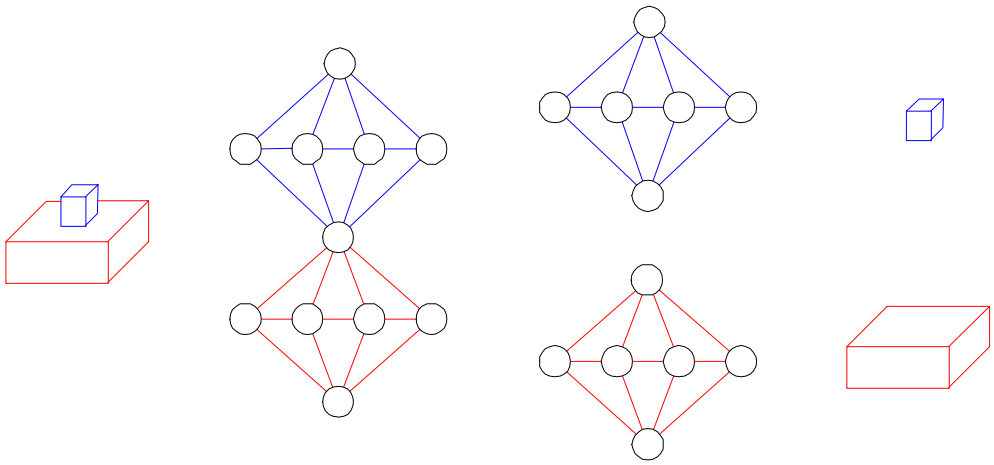


Figure 2.5 : Méthode des noeuds de séparation. À gauche, l'objet original et son graphe ; à droite, l'objet résultant de la séparation du graphe.

- Cette méthode est surtout utile pour localiser les dépressions et protrusions reliées à une seule surface de l'objet. Par exemple, une protrusion chevauchant une arête de l'objet serait plus difficilement localisable avec cet algorithme.

2.3.2 Approches par décomposition spatiale

Tel que le nom l'indique, les approches par décomposition spatiale impliquent un découpage initial de l'objet en volumes élémentaires peu signifiants, et qui seront ultimement ré-assemblés afin de reconstituer les volumes principaux de l'objet. Bien qu'il s'agisse *a priori* d'une technique de localisation de caractéristique, la phase finale d'aggrégation peut comporter un volet intrinsèque d'identification.

2.3.2.1 La décomposition cellulaire

Dans la stratégie de décomposition cellulaire, l'espace est discrétisé en éléments primitifs, de forme et dimensions régulières ou non. Les algorithmes de découpage peuvent recourir à des méthodes géométriques simples, comme l'extension des surfaces

(Dong et Wozny [40]), la coupe par plans parallèles (Tseng et Joshi [41]), ou procéder selon des techniques plus élaborées, comme la tétraédrisation de Delaunay (Sapidis et Perruchio [42]).

Parmi les avantages et inconvénients de cette méthode, nous retenons :

- Le nombre élevé de solutions possibles. Si la phase initiale de découpage demeure relativement contrôlable malgré les différentes morphologies d'objet, il devient difficile par contre de mettre au point une stratégie d'aggrégation optimale. Face à l'explosion combinatoire des solutions, la difficulté consiste à identifier et rejeter assez rapidement les interprétations non-réalisables. Citons par exemple la méthode des *chemins hamiltoniens* (Trabelsi *et al.* [43]) pour tenter de résoudre ce problème.

- Contrepartie de l'inconvénient précédent, il est plus facile d'obtenir des interprétations multiples qu'avec toute autre méthode. En effet, la plupart des algorithmes posent une hypothèse initiale, et anticipent un résultat précis qui sera seulement confirmé ou infirmé. Par nature, la décomposition cellulaire ne peut présumer aucune solution particulière, et doit nécessairement explorer toutes les solutions.

- La décomposition cellulaire impose une exigence de polyédrisme sur l'objet analysé.

2.3.2.2 La décomposition convexe

Proposée initialement par Woo [44], la décomposition convexe procède par un découpage récursif de l'objet en volumes alternativement solides et creux. La technique a été améliorée par Kim [45] afin de résoudre les problèmes de non-convergence de l'algorithme initial. L'algorithme est maintenant connu dans la littérature sous le terme ASVP, pour *Alternating Sum of Volumes with Partitioning*.

Le principe de l'ASVP est illustré à la figure 2.6. L'idée de base consiste à calculer la coquille convexe de l'objet, puis à lui soustraire l'objet d'origine ; l'algorithme reprend ensuite les mêmes étapes sur l'objet résultant, jusqu'à épuisement du matériau. Une deuxième phase d'analyse, illustrée à droite de la figure 2.6, vise à éliminer les résidus de calcul pour ne laisser que les entités volumétriques significatives.

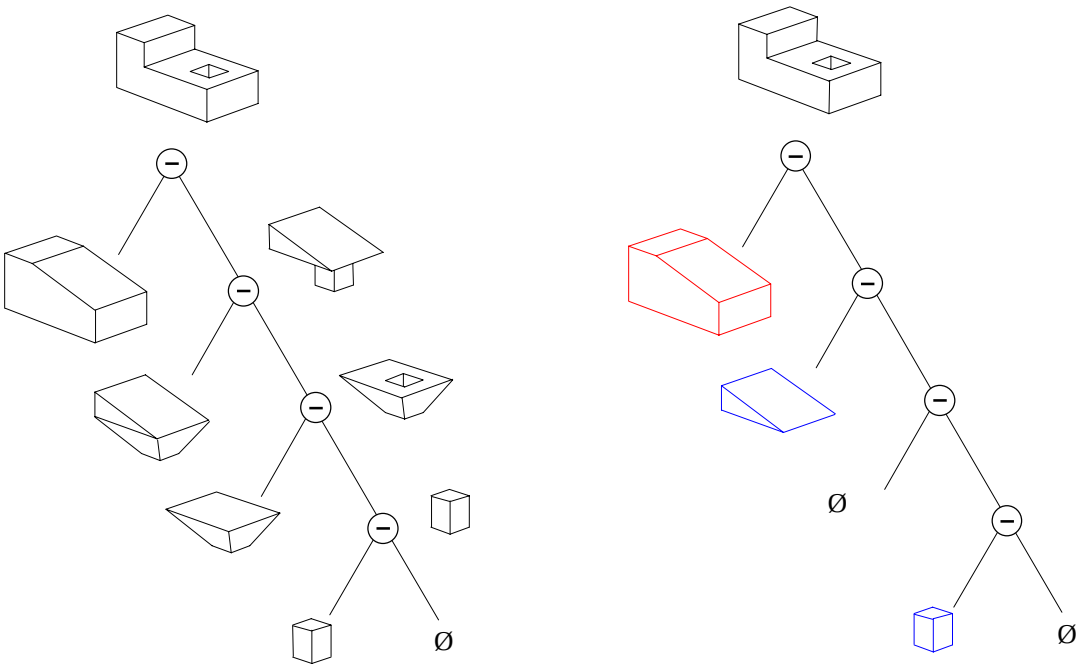


Figure 2.6 : Décomposition convexe. (Tiré de [45])

Parmi les avantages et inconvénients de l'ASVP, nous citerons :

- Outre l'extraction proprement dite des volumes, cette décomposition produit aussi une structure hiérarchique, utile pour déterminer la séquence de fabrication. Parienté et Kim [46] ont proposé dans cette foulée la *reconnaissance incrémentale*, raffinement de l'ASVP qui évite de recalculer l'arborescence complète lorsqu'un objet est faiblement modifié.

- À l'inverse de la décomposition cellulaire, la décomposition convexe produit un nombre limité de solutions ; selon Kim, l'algorithme serait optimal, au sens d'une minimisation du nombre de volumes extraits.

- Inconvénient important, le fait que les volumes obtenus ont souvent une forme assez quelconque, parfois très éloignée de l'objectif recherché. Dans le cadre de ce projet, il devient difficile d'accepter l'ASVP comme technique d'extraction des géons, puisque ceux-ci sont définis implicitement par l'algorithme plutôt que par le concepteur. Cet inconvénient de l'algorithme actuel retarde encore son utilisation dans un contexte manufacturier, malgré les travaux de Waco et Kim [47] pour convertir le résultat d'une décomposition convexe en entités manufacturables.

- Comme pour la décomposition cellullaire, les corps doivent être polyédriques. L'ASVP serait théoriquement applicable pour des surfaces non-planes (Minich [29]), mais aucun résultat n'a encore été publié à l'appui.

2.3.3 Approches symboliques

En l'absence d'un point de départ certain au sein de la description CAO, l'extraction de volumes demeure foncièrement un processus essai/erreur ; les approches symboliques (*rule based methods*) tentent justement de répondre à cette problématique par l'utilisation de techniques d'intelligence artificielle. L'outillage symbolique permet effectivement d'aller bien au-delà du seul paradigme d'acceptation ou de rejet d'une hypothèse, propre aux techniques algorithmiques.

Les stratégies mises en oeuvre sont variées, et relèvent généralement des méthodes de raisonnement incertain (*uncertain reasoning*) : placer des faits sur un tableau jusqu'à ce qu'il y ait suffisamment d'évidence pour affirmer ou réfuter une hypothèse ; traiter les hypothèses selon le degré de confiance sur le résultat ; propager

une particularité géométrique locale à travers le modèle ; accumuler des indices à partir de sources autres que la seule description CAO ; etc.

Les travaux de Vandenbrande et Requicha [48] constituent actuellement la référence en la matière ; on pourrait citer aussi Han et Requicha [49] et Regli [50] dans la même lignée.

Il faut souligner au passage que les systèmes à base de règles ne sont pas exclusifs aux approches symboliques : cet outil est utilisé en effet dans tous les travaux antérieurs. Nous ne recensons d'ailleurs aucun travail implémenté par programmation conventionnelle. Cependant, à l'extérieur du cadre des approches symboliques, les systèmes à base de règles sont rarement exploités à fond, leur mission première étant surtout de simplifier l'implémentation des différentes techniques.

2.3.4 Approches neuronales

Peu de travaux ont été publiés à ce jour concernant l'extraction de caractéristiques par réseaux de neurones. Comme on le sait, les réseaux neuroniques simulent l'activité cognitive par le biais d'éléments de traitement simples, les *neurones*, massivement interconnectés entre eux par des *synapses*. À chaque synapse est associé un poids, facteur multiplicatif qui augmente ou atténue le signal provenant du neurone source. Les neurones produisent une sortie qui dépend généralement d'une fonction non-linéaire du signal d'entrée résultant.

Prabhakar et Henderson [51] utilisent plusieurs réseaux MLP (*Multi Layer Perceptrons*) indépendants, chaque réseau étant dédié à la reconnaissance d'une caractéristique particulière. Principale différence avec le modèle usuel, les poids d'un réseau sont calculés par le concepteur, en fonction de la caractéristique à apprendre, plutôt que par présentation répétée de patrons. Un objet de n surfaces est d'abord codifié sous

forme de *matrice d'adjacence*, structure de $n \times n$ vecteurs de 8 éléments, qui résume autant la topologie que la géométrie des surfaces. La matrice est ensuite injectée ligne par ligne à l'entrée de chaque réseau, dont l'unique neurone de sortie signalera la présence ou non d'une caractéristique. Cette technique permet aussi d'identifier les surfaces qui composent une caractéristique.

Bien que la proposition de Prabhakar et Henderson représente un point de départ interessant, les résultats obtenus révèlent cependant de nombreuses lacunes. L'architecture actuelle confond par exemple certaines caractéristiques semblables, signale plusieurs fois la présence d'une même caractéristique, ou accepte certaines caractéristiques qu'elle aurait dû rejeter.

Nezis et Vosniakos [52] reprennent l'architecture MLP, mais sous sa forme conventionnelle, où un seul réseau doit détecter toutes les caractéristiques désirées. La couche de sortie contient huit neurones, aptes à représenter huit classes différentes, alors que la couche d'entrée reçoit un vecteur de 20 éléments. L'apprentissage s'effectue par rétropropagation, avec un ensemble d'entraînement relativement modeste.

Selon la méthode proposée, l'objet analysé est d'abord converti sous forme de graphe relationnel, et découpé ensuite en sous-graphes à l'aide d'heuristiques, chacun des sous-graphes représentant présumément une seule caractéristique. Les auteurs codifient aussi l'objet selon le principe de la matrice d'adjacence, à la différence que leur système génère autant de matrices d'adjacence qu'il y a de sous-graphes. Une méthode originale permet de convertir chacune des matrices, de dimensions variables, en un vecteur de taille fixe.

Si le domaine de modélisation est identique à celui de Prabhakar et Henderson, soit les objets composés de surfaces planes et cylindriques, les résultats obtenus sont

cependant plus encourageants. Tel qu'illustré à la figure 2.7, le réseau peut en principe reconnaître autant les caractéristiques usuelles que leur variantes complexes.

Les expériences menées indiquent que le réseau reconnaît avec succès la grande majorité des caractéristiques apprises, même en présence de faible interférence. Élément prometteur pour un usage industriel, l'apparente possibilité d'un apprentissage incrémental, qui permettrait à l'usager d'ajouter peu à peu les caractéristiques désirées en mémoire. La principale lacune relevée par les auteurs concerne le mécanisme de découpage en sous-graphes, qui échoue parfois dans l'extraction des bonnes surfaces.

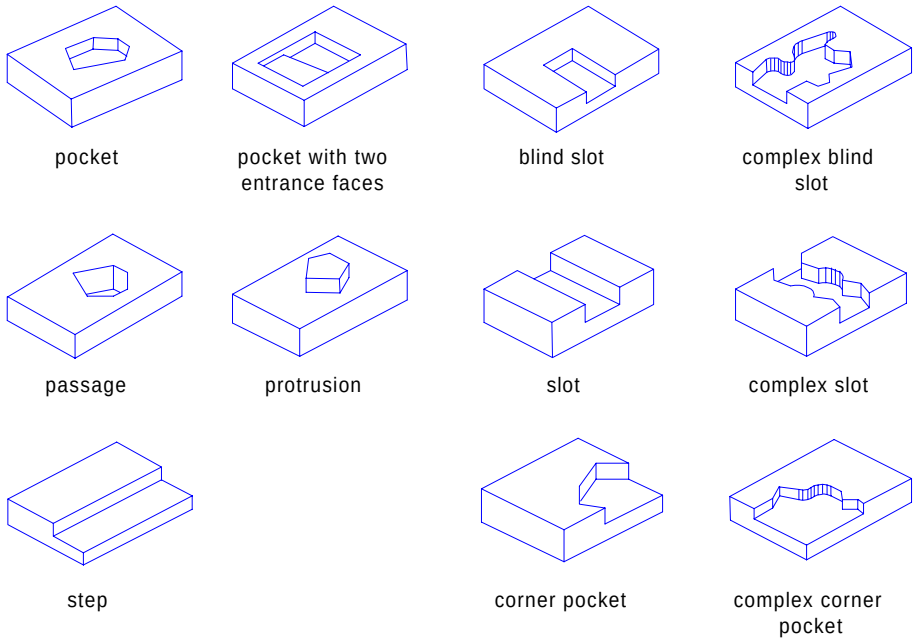


Figure 2.7 : Caractéristiques reconnues par Nezis et Vosniakos. (Tiré de [52])

Notre commentaire personnel sur ce travail concerne surtout le recours aux sous-graphes. Bien que cette approche semble nettement plus prometteuse que celle de Prabhakar et Henderson, eu égard aux seuls résultats, nous devons conclure que le succès du réseau dépend en grande partie de la bonne segmentation du graphe original

en sous-graphes. À ce titre, le réseau de Nezis et Vosniakos n'effectue en fait que la *validation* de l'hypothèse proposée par l'étage précédent ; or, le coeur même du problème d'extraction de volumes réside dans l'identification des surfaces membres, c'est-à-dire dans la génération des hypothèses. Si le réseau de Prabhakar et Henderson apparaît à prime abord comme une simple traduction dans le domaine neuronal d'un algorithme codifiable autrement, la globalité de leur approche nous semble préférable, attendu qu'un réseau neuronique devrait segmenter par lui-même l'objet complet.

En matière d'approches neuronales, mentionnons aussi les travaux de Chen et LeClair [53], qui nous concernent moins directement puisque le réseau doit plutôt déterminer la séquence de fabrication à partir de caractéristiques connues, ainsi que ceux de Wu *et al.* [54], qui cherchent à reconnaître les caractéristiques présentes dans une vue 2D.

2.3.5 Approches syntaxiques

La reconnaissance syntaxique est une méthode bien connue en reconnaissance de forme, et déjà largement utilisée en vision artificielle et en reconnaissance du langage oral ou écrit. S'inspirant de la théorie du langage, la reconnaissance syntaxique consiste à extraire les primitives du signal source, à les codifier sous forme de chaîne de caractères, et à les apparier avec des patrons prédéfinis. À la différence des méthodes isomorphiques qui requièrent une stricte identité entre le patron source et le patron cible, les méthodes syntaxiques recourent par exemple aux *grammaires à contexte libre*, qui permettent de déduire une cible variable à partir d'une source générique.

Une telle grammaire G est définie par un ensemble de symboles terminaux V_t, qui constituent les éléments primitifs du langage, de symboles non-terminaux V_n, qui représentent les patrons intermédiaires, d'un symbole de départ unique S, et de règles de production P, qui permettent de générer les patrons appartenant au langage L(G).

Citons à l'appui les travaux de Staley *et al.* [55], qui cherchaient à reconnaître les dépressions axisymétriques (obtenues par révolution) dans une base de données CAO. Cette méthode requiert l'identification préalable des arêtes longitudinales d'un trou, qui sont ensuites converties sous forme de chaîne de caractères en fonction du sens de parcours dans un plan 2D. La figure 2.8 illustre un des 13 types de trous reconnus par cette méthode.

(a) (b)

Figure 2.8 : Reconnaissance syntaxique. (a) Ensemble de primitives directionnelles de longueur unitaire, inspirées du code de Freeman. (b) Codification syntaxique d'un trou fraisé à fond plat. (Tiré de [55])

Pour l'exemple de la figure 2.8, la grammaire serait :

$G = (V_t, V_n, P, S)$ $\quad P = \{ S \rightarrow fAh,$

$V_t = \{a, e, f, g, h\}$ $\quad A \rightarrow B \mid fAh,$

$V_n = \{S, A, B, C\}$ $\quad B \rightarrow eBa \mid eCa,$

$\quad C \rightarrow g \mid Cg \}$

Le trou *x* de la figure 2.8 (b), codifié sous la forme feeegaaah, peut être obtenu par l'application successive des règles de production S, A, B et C dans l'ordre suivant :

S : $x = fAh$

A : $x = f(B)h = fBh$

B : $x = f(eBa)h = feBah$

B : $x = fe(eBa)ah = feeBaah$

B : $x = fee(eCa)aah = feeeCaaah$

C : $x = feeegaaah$

Ainsi, n'importe quel trou *x* qui peut être dérivé des règles de production P précédentes appartiendra au langage L(G), et sera donc classifié comme "trou fraisé à fond plat". On résume plus simplement le langage engendré par cette grammaire par :

$$L(G) = \{f^n e^m g^p a^m h^n \mid m, n, p = 1, 2, ...\}$$

L'instance particulière de la figure 2.8 s'exprime donc plus simplement fe^3ga^3h. La forme L(G) exprime bien d'ailleurs l'insensibilité de cette méthode aux facteurs d'échelle : par exemple, le langage engendré par G n'impose aucune dimension particulière sur *x*, sinon que les arêtes e et a, ainsi que f et h, soient de même longueur.

Cette méthode, en principe extensible à d'autres caractéristiques que les seuls trous, comporte néanmoins certains inconvénients. Qu'en est-il par exemple si le trou n'est pas perpendiculaire à la surface porteuse ? Dans l'exemple de la figure 2.8, les arêtes f et h seraient de longueurs inégales, et le trou serait rejeté. Ce même trou pourrait aussi être percé perpendiculairement à une surface oblique, et serait alors représenté par une toute autre chaîne de caractères ; la représentation n'est donc pas invariante aux rotations.

Autre problème critique, l'identification des arêtes longitudinales : dans la forme proposée, l'utilisateur doit indiquer manuellement la position du trou, de ses arêtes, et du point de départ. Les auteurs suggèrent effectivement une avenue pour l'identification automatique des arêtes, mais qui s'apparente tout à fait aux techniques de graphe. À cet égard, nous reprenons ici le même commentaire que pour Nezis et Vosniakos, à savoir que la méthode permet seulement la validation d'une hypothèse ; le problème de segmentation reste entier.

Comme autres travaux de même nature, on pourrait citer Jakubowski [56], qui classifiait un objet par analyse syntaxique des arêtes de sa silhouette. Kyprianou [23] avait pour objectif de classifier les protrusions et dépressions d'un objet. L'analyse

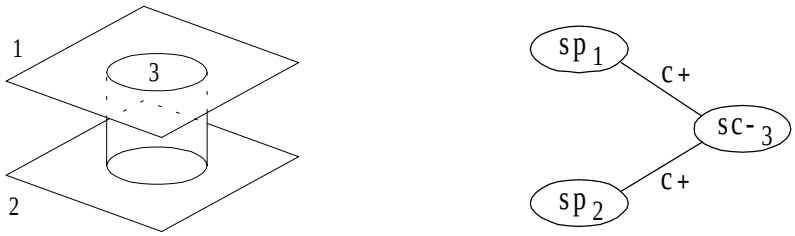
syntaxique s'appuyait sur une grammaire fondée sur les relations d'adjacence des surfaces, la concavité et convexité des arêtes, et l'existence de certaines boucles d'arêtes.

Choi *et al*. [57] généralisent la reconnaissance syntaxique dans un domaine 3D par le recours à des primitives de surfaces plutôt que des primitives d'arêtes. Par exemple, une cavité cylindrique simple serait décrite en forme Backus-Naur par la règle suivante :

HOLE ::= HSS {HES} HBS

où HSS indique une surface plane trouée (*hole starting surface*), HBS une surface circulaire de fond de trou (*hole bottom surface*), et HES une surface cylindrique (*hole element surface*). La paire d'accolade indique la répétition éventuelle de la primitive une ou plusieurs fois. Les auteurs parviennent ainsi à classifier avec un certain succès des trous, marches, rainures et dépressions, même en présence d'interactions modérées. La méthode générale requiert par contre un pré-traitement des entités, qui emprunte par exemple aux techniques de graphe, afin d'assigner les bonnes primitives syntaxiques aux surfaces.

Nous terminerons cette revue des travaux antérieurs en évoquant les *grammaires de graphe*, méthode de reconnaissance syntaxique qui repose sur la dérivation d'un graphe plutôt que d'une chaîne de caractères. L'objet doit être converti sous forme de *graphe augmenté*, où les noeuds indiquent les entités géométriques, et les arcs expriment les contraintes entre les noeuds. Les noeuds représenteront le plus souvent des surfaces, alors que les contraintes seront d'ordre topologique (adjacence, convexité d'arête, position relative des surfaces, etc.) ou géométrique (distance, angle, planarité, parallélisme, etc.). La figure 2.9 illustre un graphe de ce genre.



(a) (b)

Figure 2.9 : Graphe augmenté. (a) Portion d'objet. (b) Graphe augmenté correspondant, avec sp : surface plane, sc- : surface cylindrique concave, et c+ : contrainte de cercle convexe. (Adapté de [58])

Pour l'exemple de la figure 2.9, une caractéristique de type "passage" serait définie en langage symbolique de la façon suivante :

```
(defrule thru-hole
        (sp x)
        (sp y)
        (sc- z)
        (constraint (x z) c+)
        (constraint (y z) c+)
⇒
        (opening1-of-hole x)
        (opening2-of-hole y)
        (inside-of-hole     z)    )
```

Remarquer que cette règle est exprimée à l'inverse d'une définition proprement dite, à l'image du mécanisme de reconnaissance qui analyse les règles de production "vers l'arrière" (*bottom-up parsing*). La règle précédente pourrait d'ailleurs être utilisée dans la définition d'une autre caractéristique, plus globale, qui contiendrait un passage ; à cet égard, les grammaires de graphe permettent donc de définir un langage de formes.

Les méthodes de reconnaissance fondées sur les grammaires de graphes sont plus efficaces que les techniques isomorphiques, bien qu'elles impliquent aussi une recherche

de patrons dans le graphe [25]. Comme pour toutes les techniques de graphe, les interactions sont difficiles à traiter. Les travaux les plus souvent mentionnés dans le cadre des grammaires de graphe sont ceux de Pinilla *et al.* [59] et de Fu *et al.* [58].

2.4 Commentaires généraux

Ces différents travaux antérieurs nous inspirent quelques commentaires généraux. Le premier concerne l'évidente similitude entre les géons et les caractéristiques, qui définissent dans chaque cas un volume élémentaire d'un objet. La nuance entre les deux est surtout d'ordre fonctionnel : une caractéristique correspond généralement à un volume usinable de l'objet, alors que le géon délimite une composante perceptuelle de la pièce. À cet égard, l'un et l'autre ne sont pas foncièrement différents ; ce sont surtout leurs propriétés globales qui les distinguent.

Par exemple, une description géon est indépendante de tout ordre d'énumération, alors que les caractéristiques d'un objet impliquent une séquence précise, éventuellement dotée d'opérateurs. De plus, la description géon explicite tous les volumes de l'objet, alors que les caractéristiques ne s'intéressent qu'à quelques volumes. Mentionnons d'autre part que les géons sont plus librement définis, dans la mesure où les contraintes de manufacturabilité et d'accessibilité, propres aux caractéristiques, leurs sont étrangères.

On note d'ailleurs la prévalence des caractéristiques négatives dans tous les travaux. On comprend facilement ce phénomène, sachant que les opérations d'usinage impliquent un enlèvement de matière. Cette préférence nous oppose quelque peu à ces travaux, puisque notre projet serait plutôt attiré par les primitives solides positives de l'objet. Nonobstant ces différences, il y a une affinité très manifeste entre les algorithmes utilisés pour la reconnaissance de caractéristiques, et ceux requis par notre projet.

Quant aux méthodes utilisées à date, nous devons souligner la faiblesse générale des algorithmes de segmentation de données. En effet, plusieurs auteurs font grand cas de la technique d'identification, éventuellement assez sophistiquée, mais laissent les étapes préalables dans une navrante obscurité. Pourtant, aucune identification n'est possible sans une pré-sélection adéquate des entités ; il appert à cet égard que les techniques d'exploration préalable du modèle reposent toujours sur l'heuristique. Ceci sans parler des quelques travaux où cette pré-sélection est effectuée manuellement, et qui font complètement abstraction de cette phase difficile du processus global.

On constate par ailleurs que les graphes sont très souvent utilisés dans les travaux, seuls ou en conjonction avec d'autres techniques. Ceci n'a rien de surprenant dans la mesure où les graphes résument bien les relations topologiques au sein du modèle, et qu'ils sont faciles à implémenter. Nous tenons à souligner par contre que les graphes n'apportent généralement rien de nouveau à la connaissance de l'objet : ils constituent en effet une traduction littérale d'une information immédiatement disponible dans le domaine géométrique. À ce titre, plusieurs algorithmes dits de graphes peuvent être appliqués directement dans l'univers géométrique, sans coût de calcul supplémentaire. Le principal avantage des graphes est en fait d'ordre perceptuel, puisqu'ils procurent une représentation simplifiée de l'objet qui facilite le développement d'algorithmes.

Compte tenu du domaine d'entrée de ce projet, plusieurs techniques pourraient être utilisées pour l'extraction des géons, et plus particulièrement les approches par graphes, les approches symboliques et les approches syntaxiques. Nous ne saurions affirmer qu'une de ces méthodes soit nettement supérieure aux autres : tel que nous le verrons au prochain chapitre, l'absence d'une définition stable pour le géon rend toute approche sujette à caution.

CHAPITRE 3

MÉTHODOLOGIE

3.1 Modifications proposées à la théorie des géons

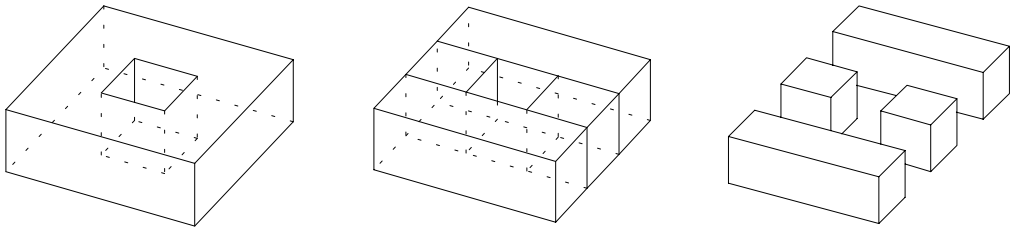
Bien que la théorie de *Reconnaissance par composantes* soit compatible avec nos objectifs généraux, on doit souligner que cette théorie avait été formulée dans un contexte de vision 2D, alors que notre projet se déroule dans un univers 3D. Chaque domaine ayant sa problématique spécifique, il nous faut donc proposer quelques ajustements à la théorie pour l'adapter à notre projet. Voici un résumé de nos propositions, qui seront détaillées dans les prochains paragraphes :

- Introduction de géons négatifs ;
- Introduction du type d'arêtes génératrices *hybrides* ;
- Simplification de l'attribut de dimension ;
- Codification de la connectivité en fonction des axes de balayage.

3.1.1 Existence de géons positifs et négatifs

L'approche par géons appliquée à des objets manufacturés entraîne rapidement le besoin de géons *négatifs*, c'est-à-dire d'entités immatérielles, dont l'existence découle de la morphologie de la pièce. Au même titre que les géons *positifs* correspondent aux constituantes solides de la pièce, les géons négatifs représenteraient les parties creuses de l'objet.

Un simple trou dans une pièce quelconque illustre bien la pertinence des géons négatifs : en termes descriptifs, il est plus simple de reconnaître une existence explicite à ce trou que de le définir indirectement par le biais des géons solides qui l'entourent. Remarquer que dans cette dernière hypothèse le trou demeurerait quand même inconnu dans la description géon, puisque aucune primitive géon ne modélise ce type d'entité. La figure 3.1 illustre ce dilemme.



(a) (b) (c)

Figure 3.1 : Justification des géons négatifs. (a) Objet à géoniser. (b) Un découpage possible de l'objet en géons. (c) Les quatre géons identifiés délimitent un trou, mais aucun n'indique explicitement sa présence.

Outre le recours aux géons négatifs, une autre avenue possible serait d'admettre l'existence de plusieurs boucles génératrices pour un même géon ; l'objet illustré à la figure 3.1 (a) appartient à cette catégorie. En termes mathématiques, il serait effectivement loisible de modéliser cet objet à l'aide d'un seul cylindre généralisé, donc un seul géon, dont le profil générateur serait formé de deux boucles carrées. Nous rejettons d'emblée cette solution, étant donné la perte de pouvoir discriminant qu'elle entraîne. En effet, l'objet de la figure 3.1 (a) et un autre semblable, mais sans le trou, auraient exactement la même description géon.

En plus d'accroître le vocabulaire descriptif, le recours aux géons négatifs permet aussi de simplifier le processus d'extraction volumétrique. Dans l'exemple de la figure 3.1, le système doit découper la pièce initiale en plusieurs morceaux, même si le parallé-

lépipède solide formait déjà un cylindre généralisé. On peut déjà anticiper la difficulté de générer les arêtes internes, entités initialement absentes de la description CAO ; de plus, l'existence de solutions multiples rend cette avenue moins attrayante. En contrepartie, le recours aux géons positifs et négatifs produit une solution unique, et requiert uniquement l'analyse des entités réelles de l'objet.

Les géons négatifs permettent un rapprochement évident avec la représentation CGS des objets. Dans ce schéma de modelisation, les corps sont le résultat d'un ensemble d'opérations d'union, d'intersection, et de différence entre des primitives volumétriques. Tel qu'illustré à la figure 3.2, le géon négatif n'est rien d'autre que le complément d'un géon positif, retranché d'un autre géon positif. Si l'immatérialité du géon négatif en fait une entité impalpable, abstraite, il est cependant perceptible par l'empreinte qu'il laisse sur une pièce.

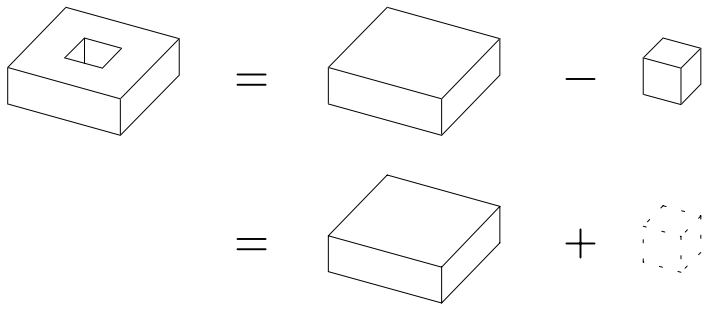


Figure 3.2 : Signification des géons négatifs.

Le recours à des géons immatériels est tout à fait nouveau dans le domaine. En effet, les *géoniciens* n'ont jamais eu à recourir à ce type d'entité dans leurs travaux, bien que les mécaniciens manipulent depuis longtemps le concept de caractéristique négative. La nature des objets habituellement reconnus dans les approches géon explique pourquoi le principe n'est jamais passé d'un domaine à l'autre.

3.1.2 Type d'arêtes génératrices

Un des attributs du géon concerne les arêtes du profil générateur. Tel que mentionné au chapitre 1, Biederman n'admettait que deux valeurs pour cet attribut, soient les profils constitués d'arêtes uniquement droites, ou d'arêtes uniquement courbes.

Une brève étude des objets soumis au programme indique que les profils générateurs de certains géons pourront être formés d'arêtes droites et courbes ; en témoigne par exemple un des bras de la pièce 3, illustré à la figure 3.3 (a). L'obligation de recourir à des géons aux arêtes uniquement droites ou courbes force le découpage de cet objet en plusieurs géons, illustrés à la figure 3.3 (b).

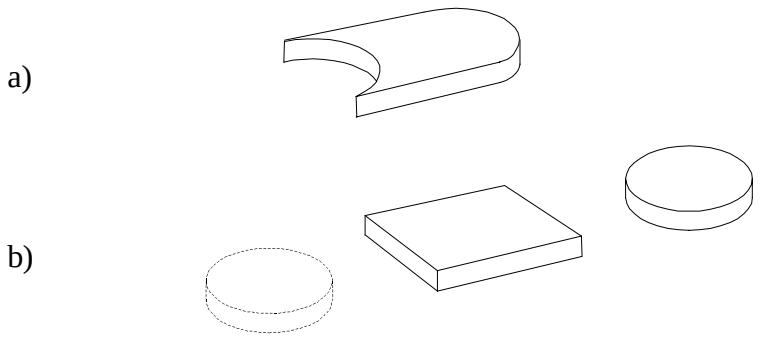


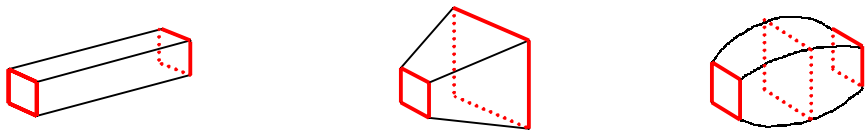
Figure 3.3 : Géonisation selon des arêtes droites ou courbes. (a) Objet à géoniser. (b) Géons obtenus. Il s'agit de géons partiellement imbriqués.

Cette solution constitue une distorsion évidente du principe de reconnaissance par composantes : trois géons ont été extraits là où l'oeil n'en percoit qu'un seul. Une telle solution serait aussi laborieuse à implémenter puisque peu d'évênements géométriques justifient un tel découpage. Ce résultat provient en fait d'une contradiction théorique : la contrainte la plus fondamentale de la RBC exige qu'un géon unique représente un cylindre généralisé unique, alors que les valeurs prévues de l'attribut "Type d'arêtes" interdisent les géons aux arêtes droites et courbes.

À la lumière de cette observation, il nous apparaît donc essentiel d'admettre une troisième valeur d'attribut pour qualifier le type d'arêtes génératrices, à savoir la valeur *hybride*, qui indiquera un profil générateur formé d'arêtes droites *et* courbes. Cette nouvelle valeur résoud la contradiction mentionnée auparavant, et permet de représenter l'objet de la figure 3.3 (a) par un seul géon, tel qu'il se doit puisqu'il s'agit d'un seul cylindre généralisé.

3.1.3 Simplification de l'attribut de dimension

On se souvient que les géons sont qualifiés par un attribut de dimension, qui exprime la taille relative du profil générateur le long de l'axe de balayage. Il s'agit en l'occurrence de l'équivalent qualitatif de la fonction de balayage ; les trois valeurs prévues pour cet attribut sont illustrées à la figure 3.4.



(a) (b) (c)

Figure 3.4 : Attribut de dimension. (a) Dimension constante. (b) Dimension croissante. (c) Dimension croissante et décroissante.

Bergevin faisait remarquer [9] que la valeur d'attribut *Croissant et décroissant* était peu utile en pratique, puisque les géons concernés peuvent être découpés en deux géons de dimension strictement croissante ; cette remarque s'applique aussi pour nous. En effet, un objet modelisé uniquement avec des primitives de surface régulières (surface plane, cylindrique, conique, sphérique, etc.) présentera toujours une arête bien définie à l'endroit où se joignent les demi-géons. La situation serait toutefois différente en présence de surfaces sculptées, puisque une seule surface NURBS suffirait à modéliser les deux demi-géons. La continuité géométrique d'ordre 2, usuelle à ce genre de

surface, rendrait plus ardue la localisation de l'arête virtuelle séparant les deux demi-géons, et justifierait probablement la valeur d'attribut *Croissant et décroissant* de Biederman.

Puisque le domaine d'entrée de notre projet ne comporte aucune pièce de dimension croissante et décroissante, d'une part, et que les surfaces sculptées sont exclues de nos primitives de modélisation d'autre part, nous proposons en conséquence d'éliminer l'attribut Croissant et décroissant, qui ne répond à aucun besoin précis pour nous. L'actuelle valeur d'attribut *Croissant* sera remplacée par la valeur *Variable*, plus générale, ce qui permet de satisfaire nos besoins immédiats de représentation, tout en laissant une marge de manoeuvre pour un éventuel élargissement du domaine d'entrée.

3.1.4 Codification de la connectivité

Les valeurs de connectivité suggérées par Biederman étaient typiques de la perception humaine : géon situé *au-dessus* d'un autre et rattaché à sa surface *courte*, etc. Ce genre d'attribut pose des difficultés d'application évidentes dans notre contexte : par exemple, si la position relative de deux géons est incontestable dans une scène visuelle, les qualificatifs de *dessus* et *dessous* perdent cependant toute signification pour un objet isolé, tel qu'il apparaît dans une description CAO, et dont l'orientation spatiale est de surcroît imprévisible. Le besoin d'un critère objectif nous incite à rejeter l'attribut de connectivité tel que proposé par Biedeman, comme l'ont fait la plupart de nos prédécesseurs en vision.

L'examen des pièces soumises en entrée suggère une classification de la connectivité en fonction de la position relative des axes de balayage des géons. À cet égard, nous identifions cinq cas particuliers, illustrés à la figure 3.5. Nous posons que l'axe de balayage d'un géon est défini entre les centroïdes des deux surfaces terminales

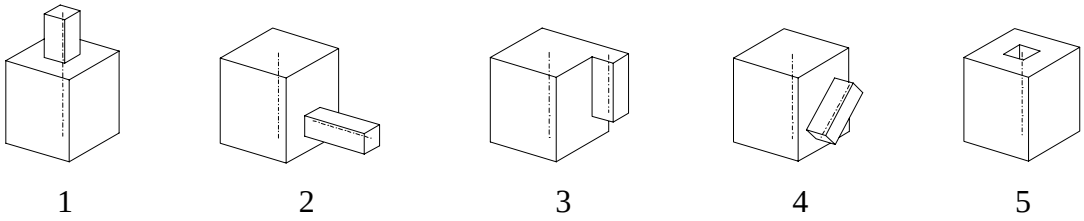


Figure 3.5 : Position relative de deux géons. On doit souligner que les cas illustrés ne représentent pas tous les cas imaginables, mais seulement les cas les plus probables.

du cylindre généralisé. Ce choix est purement pratique, dans la mesure où aucune contrainte mathématique n'impose d'emplacement précis pour l'axe de balayage d'un cylindre généralisé.

Conformément à la numérotation de la figure 3.5, les axes de balayage de deux géons peuvent être :

1. Colinéaires
2. Cosécants
3. Parallèles
4. Croisés
5. Confondus

Pour le cas des axes cosécants, il ne semble pas opportun de distinguer le type d'angle de jonction (perpendiculaire ou autre). Cette classification est fortement orientée en fonction des pièces du projet ; nous convenons cependant qu'elle pourrait poser certaines difficultés pour d'autres pièces.

Si on considère par exemple deux géons semblables à ceux de la figure 3.5 (1), mais avec le géon supérieur quelque peu incliné, il faudrait alors qualifier cette connectivité de "cosécante", puisqu'il y aurait échec du critère mathématique de colinéarité. Il y aurait là une certaine distorsion perceptuelle, puisque l'objet supposé s'apparenterait beaucoup plus à la relation illustrée en (1) qu'en (2).

Les cinq valeurs d'attribut supposent manifestement que les axes de balayages des deux géons sont droits ; cependant, pour une combinaison d'axes droit et courbe, ou de deux axes courbes, il faudrait probablement ajouter de nouvelles valeurs d'attribut, ou élargir la signification des valeurs existantes. Dans un contexte de vision 2D finalement, il pourrait s'avérer particulièrement ardu de calculer si les axes de balayage sont cosécants ou croisés[4].

Dans l'état actuel des travaux, nous ne sommes pas encore en mesure de déterminer la connectivité des géons ; notre proposition pourrait donc être modifiée sans conséquence particulière sur les travaux réalisés à date. Par exemple, pour diminuer le coût de calcul de la connectivité, une autre possibilité serait de ne tenir compte que des surfaces de contact des géons. Si on appelle T les surfaces terminales (génératrices) et L les surfaces latérales du géon, il n'y aurait alors que trois valeurs d'attribut :

1. Jonction TT : géons bout-à-bout, alignés ou non ;
2. Jonction TL : géons à angle ;
3. Jonction LL : géons côte-à-côte, parallèles ou non.

À titre d'exemple, la jonction TT correspond au cas (1) de la figure 3.5, la jonction TL au cas (2), et la jonction LL aux cas (3) et (4). Au besoin, on pourrait créer une classe distincte pour les géons concentriques ou inclusifs (par exemple une dépression excentrée dans un corps solide), ces situations s'apparentant à la jonction TT uniquement lorsque le géon négatif est un passage (plutôt qu'une dépression). L'avantage de cette proposition, c'est qu'elle requiert seulement d'identifier les surfaces de contact, alors que la proposition précédente impliquait le calcul de l'intersection des axes.

---

[4] Cette remarque n'est pas strictement théorique, puisqu'il existe dans l'environnement de notre projet un autre module de reconnaissance de géons, opérant à partir d'une image 2D. Nos choix doivent absolument satisfaire aussi les besoins de ce module si on veut espérer une convergence des descriptions géon.

Bien que cette deuxième proposition diminue quelque peu le vocabulaire descriptif, le critère essentiel demeure celui du pouvoir discriminant de l'ensemble de la description géon. À cet égard, la connectivité ne nécessite pas un grand niveau de détail, la spécificité d'une description géon reposant plus sur le nombre et la nature des géons que sur leur interrelation. Au vu des travaux antérieurs en vision, la connectivité jouerait plutôt un rôle complémentaire dans le processus d'identification des objets.

3.1.5 Synthèse des attributs

Compte tenu des modifications proposées dans les sections précédentes, un géon sera donc spécifié en fonction des cinq attributs suivants :

1. Matérialité du géon
2. Courbure de l'axe de balayage ;
3. Courbure des arêtes génératrices ;
4. Symétrie du profil générateur
5. Fonction de balayage

Les valeurs admises de chaque attribut sont mentionnées dans le tableau 3.1 ; nous y précisons en plus le nom des variables correspondantes dans nos programmes. Le répertoire complet des géons, défini par toutes les combinaisons possibles d'attributs, est énuméré à l'annexe A.

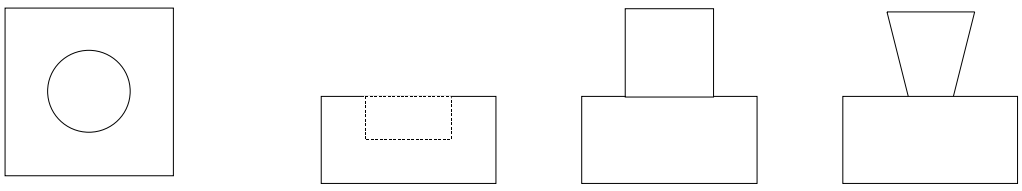
Nous avons volontairement exclu des valeurs d'attribut le zéro, afin que le module d'extraction 2D→géons puisse représenter un attribut inconnu par une valeur nulle. En effet, on peut d'ores et déjà anticiper l'impossibilité d'identifier un géon avec certitude lorsque l'angle d'observation est défavorable.

Tel qu'illustré à la figure 3.6, un simple cercle dans une image au trait ne permet de justifier aucune hypothèse particulière quant à la matérialité et la fonction de balayage du géon. Pour cet exemple, le géon central pourrait aussi bien être une dépression qu'une protrusion, cylindrique ou conique, et il n'y aurait aucune raison d'orienter le

module d'appariement vers une solution plutôt qu'une autre. Bref, un géon représenté par certains attributs nuls ne correspond à aucun géon précis, mais désigne implicitement plusieurs géons. Il est clair que les descriptions géons qui émanent de notre module ne contiennent aucun attribut indéterminé puisqu'elles constituent la référence du système, et qu'elles proviennent de données CAO complètes.

Attribut	Nom	Valeurs	Signification
Matérialité	MaterialType	+1	Géon positif
		–1	Géon négatif
Courbure de l'axe	AxisType	1	Droit
		2	Incurvé
Arêtes génératrices	EdgeType	1	Droites
		2	Courbes
		3	Hybride
Symétrie du profil générateur	Symmetry	1	Asymétrie
		2	Réflective
		3	Réfl. et rotationnelle
Fonction de balayage	SweepFunction	1	Constante
		2	Variable

Tableau 3.1 : Valeurs d'attributs intrinsèques.



(a) (b)

Figure 3.6 : Attributs indéterminés. (a) Objet observé dans l'image 2D. (b) Différentes interprétations du même objet, vu de côté.

L'attribut de connectivité pourra prendre cinq valeurs, mentionnées dans le tableau 3.2. Nous assumons ici la première proposition de codification évoquée au § 3.1.4, la conception du module d'appariement étant déjà amorcée sur cette prémisse.

Valeur d'attribut	Position des axes	Type de jonction
1	axes colinéaires	géons bout-à-bout
2	axes cosécants	jonction en T ou en L
3	axes parallèles	côte-à-côte, inclusif, etc.
4	axes croisés	jonction en X
5	axes confondus	géons concentriques

Tableau 3.2 : Valeurs de l'attribut de connectivité

Le produit du nombre de valeurs d'attributs intrinsèques indique que notre répertoire de géons est constitué de $2 \times 2 \times 3 \times 3 \times 2 = 72$ primitives génériques. En faisant abstraction des géons négatifs qui dédoublent toutes les entités solides, notre répertoire a donc la même étendue que celui de Biederman, qui comprenait 36 géons.

Une question essentielle en la matière demeure celle de la *suffisance descriptive* : notre répertoire est-il trop vaste, trop restreint, ou de taille acceptable ? Cette question demeure éminemment liée au nombre d'attributs ainsi qu'aux valeurs d'attributs admises, et implique à la source le type même d'objets à reconnaître.

En pratique, on observe que le domaine manufacturier recourt fréquemment à un petit nombre de primitives solides, parmi lesquelles figurent le parallélépipède, le cylindre et le cône, et très peu à d'autres primitives, tels les géons incurvés ou asymétriques[5]. Comme nous le verrons plus loin, les dix pièces expérimentales de ce projet

[5] Encore que ceci dépend du domaine d'application considéré, donc des opérations manufacturières admises. Par exemple, les tores et les sphères sont courants en moulage.

n'utilisent en effet que 11 des 72 géons possibles, ce qui porte à croire que notre répertoire est nettement surdimensionné. Deux facteurs viennent cependant pondérer cette présomption, à savoir :

- Compte tenu de l'infinité d'objets manufacturés possibles, un échantillon de dix pièces ne saurait être statistiquement valable pour conclure sur le nombre de classes ;

- Les pièces expérimentales ne sont pas nécessairement représentatives de l'ensemble des objets manufacturés courants. Bien qu'elles aient été conçues dans cet esprit, aucune démarche n'a été entreprise pour valider leur représentativité réelle.

A l'instar de la totalité des travaux antérieurs avec les approches géon, la pertinence du choix des attributs demeure sujette à de seules supputations théoriques, faute de pouvoir appliquer le test véritable, celui de l'application industrielle.

Remarquons finalement que notre approche suppose que les géons demeurent *sans dénomination*. Bien qu'il soit naturel de vouloir associer un descripteur linguistique précis à chaque géon (comme "cylindre", "pyramide", etc.), cette méthode est difficilement applicable pour des objets de forme quelconque, laborieuse à implémenter, et totalement superflue en terme de reconnaissance de forme.

En effet, un objet peut être classifié uniquement en fonction des attributs de ses géons, puisque chaque combinaison d'attributs détermine une classe précise. Ceci n'exclut cependant pas l'utilisation d'étiquettes génériques : dans une classification par attributs, le module d'appariement risque effectivement d'être surchargé par la manipulation de plusieurs descripteurs pour chaque géon. La solution la plus simple consiste évidemment à associer une étiquette globale à chaque combinaison d'attributs (de 1 à 72 par exemple), quelle que soit l'instance spécifique de cylindre généralisé. Rappelons à cet égard qu'une combinaison donnée d'attributs peut être instanciée en pratique par une multitude de cylindres généralisés, tel qu'illustré à la figure 1.6 (p. 15).

3.2 Définition fonctionnelle des géons

À ce stade de notre réflexion, il reste encore à définir la *loi de géonisation* des objets, qui précisera la logique de découpage d'un objet en géons. En effet, bien que les attributs intrinsèques soient nécessaires pour *classifier* un géon donné, ils ne permettent cependant pas de déduire de quels géons se compose un objet.

Sans doute est-il pertinent de souligner que nous ne cherchons pas pour l'instant à déterminer *comment* extraire les géons d'un objet. En effet, avant même de pouvoir proposer un quelconque algorithme d'analyse de données CAO, encore faut-il connaître la solution à laquelle devra parvenir cet algorithme pour une pièce. L'étape de définition fonctionnelle consiste justement à définir la solution théorique, c'est-à-dire la sortie anticipée du système, pour n'importe quelle pièce donnée.

On doit remarquer à cet égard que les géons ne sont pas strictement définis : le géon est avant tout une entité perceptuelle, qui n'est assortie d'aucune formalisation mathématique précise. La théorie de *Reconnaissance par composantes* ne prévoit effectivement aucune équation ou algorithme particulier pour identifier les géons d'un corps quelconque ; il s'agit d'ailleurs d'une lacune importante de cette théorie. Il appartient donc à l'application concernée de définir elle-même le type de géons désirés, d'une part, et de préciser les contraintes fonctionnelles qui permettent de les situer dans un objet, d'autre part.

Bien que la formulation d'une loi de géonisation semble *a priori* triviale, cet exercice se heurte rapidement au problème de la *multiplicité descriptive*. En effet, la géométrie des objets manufacturés est telle qu'il est souvent malaisé d'établir leur description géon sans conteste sur la seule base du critère perceptuel : il existe généralement plusieurs segmentations possibles d'un même objet, qui dépendent du point de départ choisi et de l'ordre dans lequel les critères de découpage sont appliqués.

On doit noter par ailleurs que la contrainte des cylindres généralisés est une condition nécessaire mais non suffisante pour définir un géon ; en témoigne la figure 3.7, qui illustre deux découpages possibles d'une des pièces expérimentales. Les géons obtenus dans chaque cas sont tout à fait valides, ce qui démontre que la contrainte des cylindres généralisés ne permet pas non plus d'obtenir une solution unique à elle seule[6].

Bien que le domaine semble se prêter on ne peut mieux à une formulation mathématique ou algorithmique d'une loi de géonisation, il s'avère extrêmement ardu en pratique de définir des critères infaillibles de découpage. Trop d'éléments géométriques, topologiques et morphologiques agissent simultanément et subtilement, entraînant rapidement cet exercice dans une interminable analyse de cas particuliers. D'autres avant nous ont pu constater la difficulté de résumer en quelques termes, ne serait-ce que verbalement, la complexité du domaine tridimensionnel.

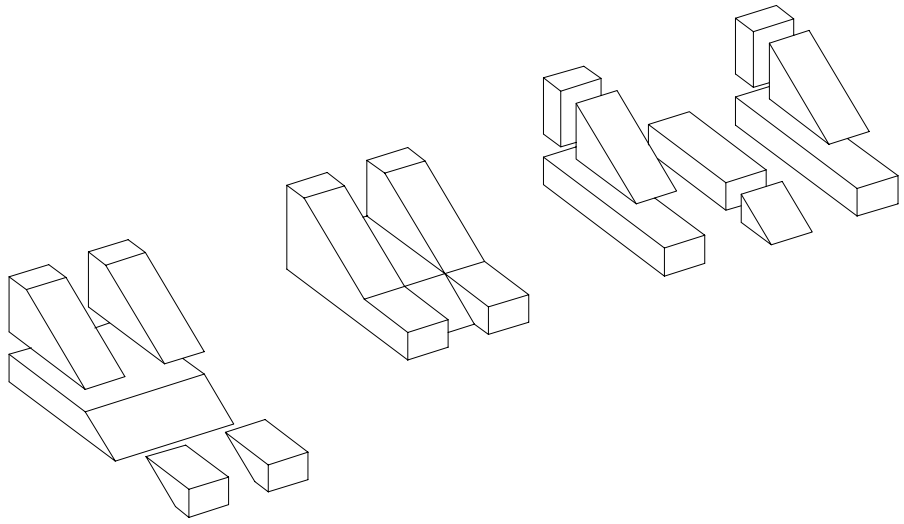


Figure 3.7 : Multiplicité descriptive des cylindres généralisés.

---

[6] Notre lecteur aura compris que le projet global de métrologie industrielle refuse les descriptions multiples d'un même objet, qui eut été une alternative possible au problème de définition d'une loi stricte de géonisation.

Face à la difficulté de formuler une loi de géonisation, nous avons examiné chacune des pièces expérimentales, et décidé *a priori* quels géons doivent en être extraits, sans formaliser aucun critère de découpage ; la loi de géonisation devient donc implicite. Rappelons qu'il s'agit uniquement à cette étape de *définir* les géons d'un objet, nonobstant la méthode utilisée par la suite pour parvenir à cette solution.

De façon générale, trois critères dictent le choix d'une description géon pour un objet donné :

1. La facilité d'extraction pour chacun des modules. Il est effectivement impératif que les géons choisis puissent être identifiés avec un coût de calcul raisonnable, autant par le module CAO→géons que le module 2D→géons, qui reçoit en entrée l'image 2D de l'objet. À cet égard, notre étude indique que parmi plusieurs solutions possibles, il faut préférer celle qui comporte le moins d'opérations de découpage sur un volume donné, c'est-à-dire qui satisfait le plus immédiatement la contrainte des cylindres généralisés. Outre qu'elle diminue le nombre de solutions, cette méthode évite aussi le découpage en sous-volumes, plus problématique à implémenter. À l'appui, notre lecteur pourra comparer les géons effectivement retenus pour la pièce 1, à la figure 3.8, avec ceux proposés pour la même pièce à la figure 3.7.

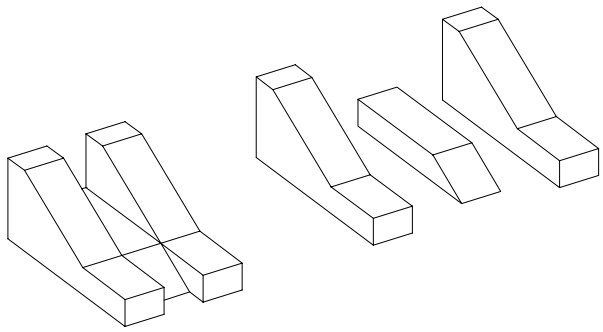


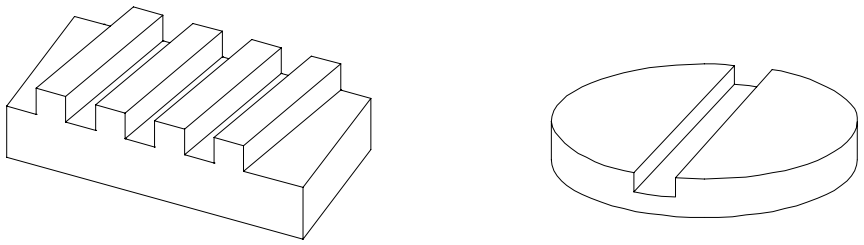
Figure 3.8 : Géonisation par découpage minimal.

Similairement, il s'avère plus aisé de rechercher surtout des géons positifs, en principe plus faciles à extraire que les géons négatifs. En effet, les géons négatifs sont souvent définis indirectement, par le biais des entités adjacentes, alors que les géons positifs reposent nécessairement sur les entités réelles de l'objet. À l'égard des géons négatifs, nous n'admettons pour l'instant que ceux situés strictement à l'intérieur d'un volume solide, tels les passages et dépressions, et excluons tout géon négatif situé en périphérie d'un géon solide, tels les rainures, les encoches et les marches.

Nous mentionnerons aussi que nos géons sont exclusifs les uns aux autres, c'est-à-dire que nous n'admettons pas de géons partiellement imbriqués, sauf si l'un est positif et l'autre négatif. Compte tenu des pièces expérimentales, nous n'avons effectivement pas eu à recourir à des géons imbriqués.

2. La puissance d'indexation dans la base de données. Il s'agit d'un critère fondamental pour ce projet : une certaine description géon permet-elle de retracer plus efficacement l'objet dans la base de données qu'une autre description géon du même objet ? Il est difficile ici de justifier avec certitude un choix plutôt qu'un autre, d'une part parce que la base de données actuelle est composée d'un nombre statistiquement insuffisant d'objets, et d'autre part parce que nous ignorons à ce jour quelle sera la stratégie d'appariement entre les géons observés dans l'image et les géons de la base de données. Cependant, on perçoit aisément qu'un nombre trop élevé de géons pour un objet augmente le temps de fouille dans la base de données, alors que trop peu de géons accroissent le risque de descriptions semblables entre objets différents. Biederman postulait à ce propos que trois géons suffisent généralement pour identifier un objet unique [4], hypothèse dont l'exactitude demeure sujette au degré de similarité entre les pièces.

3. La cohérence des descriptions géons. Si la méthode retenue impose au préalable de définir la solution désirée pour chacune des pièces disponibles, il ne doit cependant pas y avoir de contradiction entre les découpages choisis. Le respect d'une certaine logique de découpage impose donc le rejet de certaines solutions, devenues impossibles à justifier par la suite en fonction des choix antérieurs. Par exemple, si on rejette la présence de rainures sur la pièce 8, illustrée à la figure 3.9 (a), il serait difficile après coup de justifier l'existence d'une rainure au milieu de la pièce illustrée en 3.9 (b), nonobstant que cette rainure soit perceptuellement plus naturelle, ou plus facile à extraire dans une image 2D.



(a) (b)

Figure 3.9 : Contrainte de cohérence. L'absence de rainures en (a) implique aussi que l'objet (b) en soit dépourvu, à moins de trouver un critère qui justifie l'exception, qui permette de programmer ce cas particulier.

Similairement, il sera impossible par la suite de décider arbitrairement du découpage d'une nouvelle pièce. Pour éviter toute incohérence avec les modèles déjà appris, il faudra nécessairement soumettre une nouvelle pièce aux programmes implémentés pour vérifier quels sont les géons reconnus et non-reconnus.

3.3 Description géon des pièces expérimentales

Les descriptions présentées ici constituent le résultat de la définition fonctionnelle des géons ; ce sont les descriptions géon auxquelles le module d'extraction CAO→

géons devra parvenir lorsque les programmes seront au point. On doit remarquer que d'autres descriptions géon auraient été possibles pour certaines pièces, et qu'il n'y a aucune garantie absolue que l'analyse de l'image 2D conduise systématiquement aux solutions indiquées. Il est manifeste que l'interprétation d'une image dépend fortement de l'angle d'observation de la pièce.

3.3.1 Description sommaire

Nous présentons au tableau 3.3 les géons constitutifs des pièces expérimentales, suivi d'un inventaire des 11 classes de géons utilisées. Afin d'alléger le texte, les géons sont indiqués par leur identificateur ID, tel qu'indiqué à l'annexe A. L'ordre d'énumération des géons est totalement arbitraire.

Pièce	Géons constituants									
pièce 1	1	1	1							
pièce 2	5	15	15	47	47					
pièce 3	11	11	15	15	47	47	51			
pièce 4	13	47	47	51						
pièce 5	11	11	11	11	11	15	15	15	15	48
pièce 6	11	11	11	12	12	17	47	47	47	47
pièce 7	5	6								
pièce 8	5	5	5	5	5					
pièce 9	11	12								
pièce 10	11	11	11	11	11	11	11	12		

Tableau 3.3 : Géons constituants des pièces expérimentales.

ID	Nature
1	forme positive asymétrique à arêtes droites
5	parallélépipède positif
6	forme positive à dimension variable
11	cylindre positif
12	cône positif
13	forme hybride positive asymétrique
15	forme hybride positive à symétrie réflective
17	forme hybride positive à symétrie totale
47	cylindre négatif
48	cône négatif
51	forme hybride négative à symétrie réflective

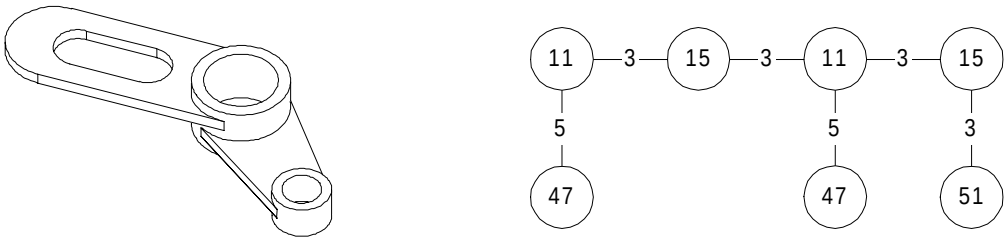
Tableau 3.4 : Inventaire des géons utilisés

3.3.2 Description complète

Afin de mieux apprécier l'information de connectivité des descriptions géons, les pièces expérimentales sont décrites ici sous forme de graphes relationnels. Les noeuds de chaque graphe indiquent l'étiquette ID du géon, alors que les arcs expriment la valeur de l'attribut de connectivité, tel qu'indiqué au tableau 3.2 (p. 72). Pour améliorer la compréhension, nous illustrons côte-à-côte la pièce réelle et sa description géon.

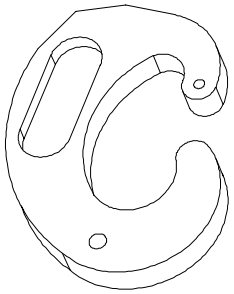
pièce 1

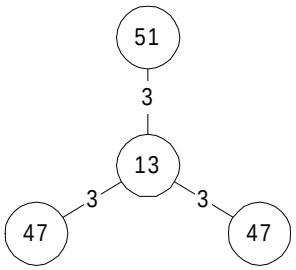
pièce 2



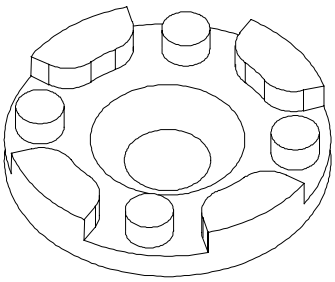
pièce 3

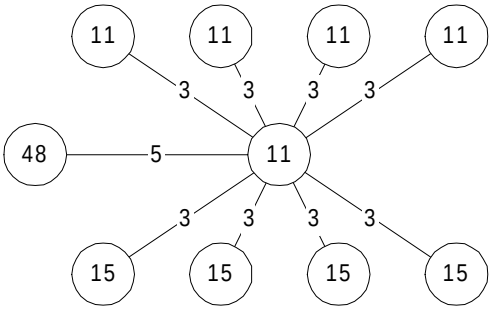
Figure 3.10 : Descriptions géon des pièces expérimentales (début)



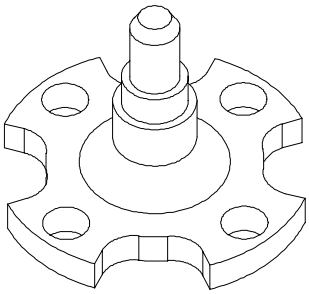


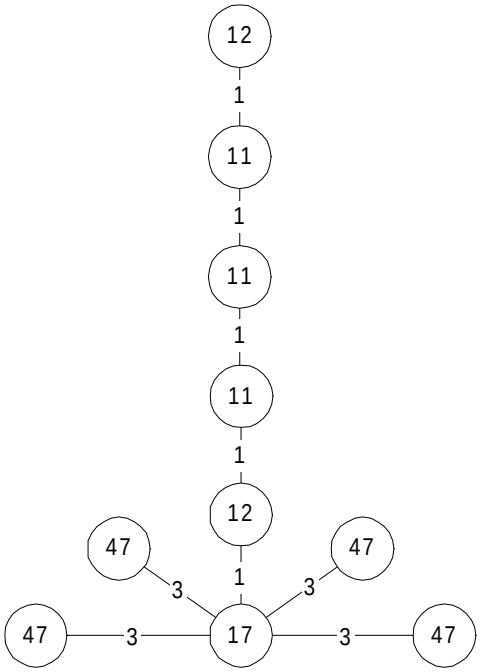
pièce 4





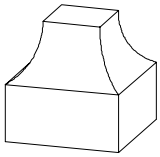
pièce 5

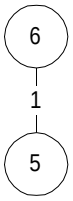




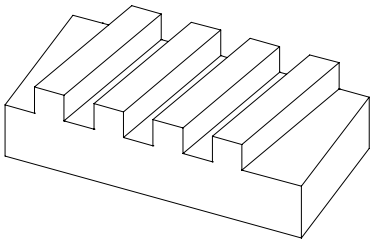
pièce 6

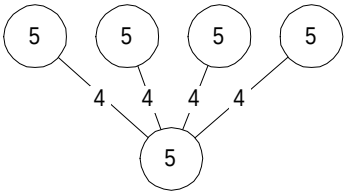
Figure 3.10 : Descriptions géon des pièces expérimentales (suite)



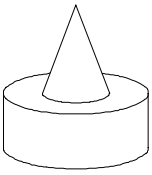


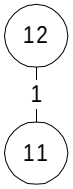
pièce 7



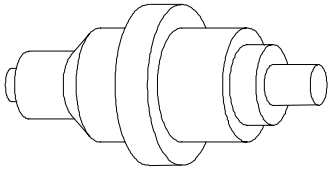


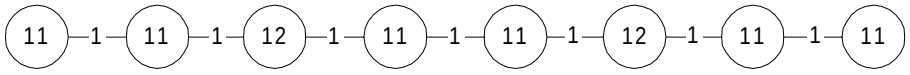
pièce 8





pièce 9





pièce 10

Figure 3.10 : Descriptions géon des pièces expérimentales (fin)

3.4 Méthodologie d'extraction des géons

3.4.1 Etapes générales

Le fonctionnement général de nos programmes est illustré à la figure suivante, et commenté par la suite. On notera au demeurant que, malgré ses particularités, ce cheminement est typique de l'ensemble des programmes d'analyse de données CAO.

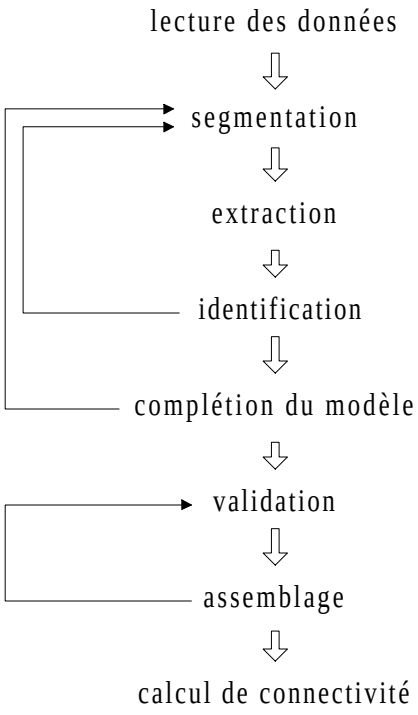


Figure 3.11 : Étapes générales d'analyse des données.

La toute première étape réside bien sûr dans la lecture des données CAO, qui doivent être converties et mémorisées sous une forme pertinente pour l'analyse géométrique ultérieure. Nous aurons l'occasion de revenir plus en détail sur ce point au prochain chapitre.

Vient ensuite l'étape la plus critique du problème, celle de la segmentation, qui consiste à rechercher dans la description CAO l'amorce d'un géon potentiel. À ce stade, l'ordinateur cherche donc à formuler une hypothèse de géon, hypothèse qui doit être justifiée par un minimum d'indices géométriques ou topologiques.

Tel que l'ont mis en évidence les travaux antérieurs, aucun algorithme ne permet de localiser avec certitude un quelconque point de départ parmi les données : différentes heuristiques doivent donc être développées, chacune s'appliquant à un type particulier de situation géométrique. Notre étude indique d'ailleurs qu'il faudra recourir à *plusieurs* techniques, éventuellement très diverses, afin d'exploiter adéquatement les propriétés du domaine.

À ce stade de l'analyse, l'ordinateur dispose seulement d'un point de départ dans le modèle ; l'extraction consiste alors à explorer le voisinage de ce point de départ, à la recherche des entités susceptibles d'appartenir au géon. Les entités sélectionnées sont mémorisées dans une structure de type *géon*, qui délimitera l'espace d'analyse de la prochaine étape.

L'identification vise essentiellement à valider l'hypothèse initiale, par le calcul des cinq attributs intrinsèques ; l'hypothèse sera considérée valide si tous les attributs ont pu être calculés avec succès. Le cas échéant, les entités appartenant au géon seront éliminées de la description CAO et mémorisées séparément. Cette procédure a pour but de réduire progressivement la taille et la complexité de l'espace d'analyse, tout en évitant l'élimination complète des entités. Le calcul de connectivité et l'activité de visualisation graphique requièrent en effet que toute l'information CAO soit préservée.

Tel que nous le verrons au prochain chapitre, un échec d'identification peut découler d'une hypothèse partiellement invalide ; le cas échéant, l'hypothèse est alors sommairement reconsidérée, et éventuellement rejetée ou acceptée une fois pour toutes.

Une procédure plus élaborée aurait exploré l'hypothèse plus à fond, jusqu'à épuisement des possibilités découlant de cette source. Nous évoquons bien sûr le principe de la fouille en profondeur, que nous n'avons pas retenu en vertu de sa complexité de mise en oeuvre dans un cadre algorithmique.

À moins que l'hypothèse n'ait été rejetée, reste alors à compléter le modèle de sorte à préserver sa cohérence géométrique. L'étape précédente implique en effet que certaines entités soient carrément enlevées de la description lorsqu'elles appartiennent à un géon reconnu, ce qui engendre de nombreuses incohérences : surfaces bordées par des boucles ouvertes, volumes non-fermés, etc. L'étape de complétion de modèle vise donc à retrouver la validité géométrique de l'objet, condition nécessaire pour poursuivre l'analyse lors des prochaines itérations.

En pratique, il s'agit surtout de générer les surfaces et arêtes manquantes sur l'objet, aux endroits où des géons ont été extraits, mais aussi de remettre à jour les pointeurs vers les entités de plus bas niveau, et de recalculer certaines informations géométriques et topologiques.

Remarquons par ailleurs que la complétion de modèle vise plus fondamentalement à mettre en évidence les géons restants sur l'objet ; à l'appui, nous référons notre lecteur à la figure 2.2 (p. 45), où une nouvelle rainure était mise à jour suite à cette opération. Après identification d'un géon négatif par exemple, l'ordinateur doit conceptuellement "remplir" le trou : il y aura donc une *évolution* de l'objet, ayant pour but d'éliminer les éléments qui empêchent la reconnaissance du géon solide.

Tel qu'illustré à la figure 3.11, l'ordinateur doit recommencer le processus de segmentation tant qu'il reste des entités ; le processus d'extraction de géons se termine naturellement lorsqu'il ne reste plus aucune entité à analyser.

Les deux prochaines étapes, de validation et d'assemblage sont plus lointaines dans notre plan de travail ; nous les mentionnons surtout pour la forme, leur besoin n'étant pas encore confirmé. Selon les algorithmes de segmentation mis en oeuvre, il est possible que les géons obtenus soient valides (i.e. que ce soient bel et bien des cylindres généralisés), mais qu'ils ne correspondent pas aux géons désirés. Une première avenue serait d'ajouter plus d'intelligence aux algorithmes de segmentation, de sorte qu'ils retournent immédiatement les "bons" géons ; l'autre possibilité serait justement de revérifier tous les géons à la fin du processus, quitte à réassembler certains d'entre eux pour obtenir des géons plus satisfaisants.

L'ensemble du processus se termine par le calcul de la connectivité, qui n'est possible que lorsque tous le géons ont été extraits de l'objet.

3.4.2 Choix d'une méthode de segmentation

La première question à résoudre pour amorcer l'analyse est de chercher quoi, et comment. En termes simples, la description CAO est formée d'une énumération de surfaces, d'arêtes et de sommets, qu'il faut explorer et analyser de façon méthodique ; face à la multitude de cas particuliers, reste encore à déterminer une stratégie globale qui permette d'identifier un nombre appréciable de géons, tout en préservant un coût de calcul raisonnable.

Il semble assez évident que l'analyse doit s'appuyer en première approche sur les surfaces, entités immédiatement inférieures aux volumes. Le principe très général consiste effectivement à agglomérer les bonnes surfaces ensemble, de sorte qu'elles correspondent à l'un ou l'autre des 72 géons admis. Les entités inférieures aux surfaces, c'est-à-dire les arêtes et les sommets, serviront la plupart du temps à réduire la complexité des calculs géométriques, puisqu'elles résument les points critiques des surfaces, mais dans un univers de dimensionnalité réduite. À l'inverse des travaux en

vision, nous n'avons cependant pas à retrouver les surfaces à partir des entités inférieures, puisque les une et les autres sont déjà explicites dans la description CAO.

Parmi les approches possibles, on doit constater que la recherche directe des primitives parmi les données serait tout à fait inappropriée. En effet, cette méthode qui relève du paradigme de *template matching* serait extrêmement onéreuse à mettre en oeuvre, puisqu'il faudrait rechercher les 72 géons les uns après les autres, que ce soit dans le domaine géométrique, ou dans un domaine transformé, tel par exemple le domaine des graphes ou des chaînes syntaxiques. Outre le problème du nombre élevé de primitives qu'il faudrait codifier et rechercher, on doit aussi souligner que cette méthode est vouée à l'échec dans un très grand nombre de cas, puisque l'assemblage de plusieurs géons modifie ou fait disparaître les entités mitoyennes, qui participent pourtant à la définition de la primitive. Les travaux antérieurs ont clairement illustré la difficulté de codifier les conditions minimales d'existence d'une primitive juxtaposée avec d'autres primitives.

Les difficultés engendrées par l'approche énumérative suggèrent de localiser les différents géons de l'objet par satisfaction de contraintes plutôt que par appariement de primitives. Il s'agirait donc d'adopter une définition générique, valable pour n'importe quel géon, plutôt que de préciser la configuration spécifique de chaque instance admise. À ce titre, nous soutenons que la seule primitive qui puisse être raisonnablement recherchée dans la description CAO est le cylindre généralisé, qui est la partie commune à tous les géons.

Ayant répondu à la question "chercher quoi", reste encore à déterminer le comment. Il nous faut donc déterminer un critère relativement fiable, qui indiquerait la présence possible d'un cylindre généralisé à un certain endroit de la description ; il appartiendra ensuite à l'ordinateur de confirmer ou infirmer l'existence d'un géon à cet

endroit. L'examen des pièces expérimentales suggère assez rapidement une méthode simple, soit le découpage de l'objet sur les boucles internes.

En effet, nous avons observé au chapitre 1 qu'une boucle interne est nécessairement le lieu de jonction entre deux géons, dont l'un est positif s'il s'agit d'une protrusion, ou négatif s'il s'agit d'une dépression. La figure 3.12 illustre à l'appui les boucles internes d'une des pièces expérimentales.

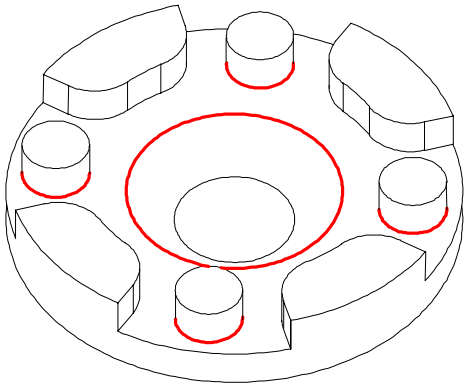


Figure 3.12 : Segmentation sur boucle interne.

Considérant que sept pièces expérimentales sur dix présentent au moins une, et souvent plusieurs, boucles internes, nous proposons donc comme stratégie générale d'identifier les boucles internes au sein de la description, puis d'amorcer le mécanisme d'hypothèse à partir de celles-ci. Remarquons qu'il s'agit fondamentalement de la méthode des *noeuds de séparation* telle que suggérée par Gavankar et Henderson [39], à la différence que nous effectuons l'opération directement dans le domaine géométrique plutôt que dans le domaine des graphes.

En cas d'échec d'identification, le programme aura minimalement l'avenue de découper l'objet sur la boucle interne, c'est-à-dire de dissocier l'objet initial en deux

morceaux. Cette méthode de recouvrement permet ainsi de faire évoluer un tant soit peu le modèle vers l'objectif désiré, en départageant les entités en deux groupes distincts.

On doit remarquer d'autre part que la procédure générale que nous prévoyons suppose implicitement l'extraction du géon de base en dernier lieu (*root last*)[7]. Plusieurs algorithmes amorcent effectivement leur raisonnement à partir du volume brut (*root first*), puis identifient peu à peu les géons portés par un principe soustractif ; dans notre cas, nous cherchons plutôt à reconnaître les géons accessoires en premier et à épurer progressivement l'objet de ses artefacts, jusqu'à obtention du géon central.

Notre dernière remarque concerne l'outillage informatique utilisé : la presque totalité des travaux antérieurs dans le domaine recouraient aux systèmes experts, alors que nos programmes sont tous écrits en C. Ce choix est effectivement questionnable, dans la mesure où l'approche symbolique présente un net avantage pour la manipulation d'hypothèses.

Nous avons perçu assez tôt dans notre étude la pertinence des systèmes experts pour résoudre ce genre de problème. Cependant, au moment d'amorcer nos travaux et de planifier nos algorithmes, aucune coquille importante de système expert n'était encore fonctionnelle dans notre laboratoire. Le temps requis pour étudier, choisir, acquérir et installer une coquille commerciale étant incompatible avec les délais de réalisation du projet, nous avons donc opté pour la voie algorithmique, ce qui était évidemment une solution par défaut.

---

[7] Dans notre terminologie, le *géon de base* est l'élément volumétrique positif qui sert de "support physique" aux autres géons ; au niveau du graphe des géons, le géon de base sera vraisemblablement le noeud central d'une topologie en étoile, ou à tout le moins, le noeud le plus interconnecté du graphe. Nous admettons aussi le terme de *géon porteur* pour signifier le cas plus général d'un géon qui en supporte un autre.

CHAPITRE 4

ALGORITHME D'EXTRACTION DE GÉONS

4.1 Environnement de développement

Nous avons développé tous nos programmes dans l'environnement de programmation KBVision. Dans une certaine mesure, cet environnement est loin d'être idéal pour notre projet, puisqu'il est dédié à la vision artificielle et qu'il ne comporte aucun outil spécifique de manipulation d'entités CAO.

Ce choix découle en fait d'un certain nombre de considérations, dont la plus déterminante est celle de l'outillage logiciel disponible. Bien que notre laboratoire d'accueil semble pauvrement équipé, il faut remarquer qu'il n'existe à ce jour aucun environnement dédié à la reconnaissance de forme dans une description CAO. Outre les nombreux modélisateurs disponibles sur le marché, on trouve aussi certaines bibliothèques de fonctions géométriques, mais aucun de ces produits ne concerne réellement l'interprétation de données CAO.

Quant à l'absence d'une plate-forme de développement plus appropriée, elle s'explique par la nouveauté du sujet pour notre laboratoire. On constate en effet que beaucoup de publications émanent de laboratoires où l'extraction volumétrique est un axe de recherche important, et qui ont développé avec les années un outillage adapté à cette problématique. Dans la mesure où notre projet constitue une première tentative dans ce domaine, on comprend mieux l'absence de ressources spécialisées au LIVIA.

KBVision n'en demeure pas moins un produit acceptable pour le développement, d'autant plus qu'il pourra accueillir la plupart des modules du projet global de métrologie industrielle, dans lequel s'insèrent nos travaux. Il s'agira d'un environnement de choix pour le développement du module d'extraction de géons à partir d'une image 2D.

* * *

En pratique, les travaux réalisés se résument à trois programmes. La figure 4.1 illustre l'enchaînement de ces différents programmes, que nous présentons très sommairement afin de donner une vue d'ensemble de notre projet :

- Une première tâche, dite *NeutralToTks*, reçoit en entrée la description CAO produite par le modélisateur, qu'elle convertit sous une forme compatible avec les mécanismes de représentation de KBVision.

- Vient ensuite le programme essentiel, *InnerLoopGeons*, qui extrait les géons selon la méthodologie exposée au chapitre 3. Tel que mentionné auparavant, d'autres programmes devront s'ajouter en aval de cette tâche, l'algorithme implémenté ne parvenant pas à extraire tous les géons.

- Une dernière tâche, *TksToSat*, convertit la description CAO résultante sous forme de fichier .sat, pour permettre la visualisation des géons extraits.

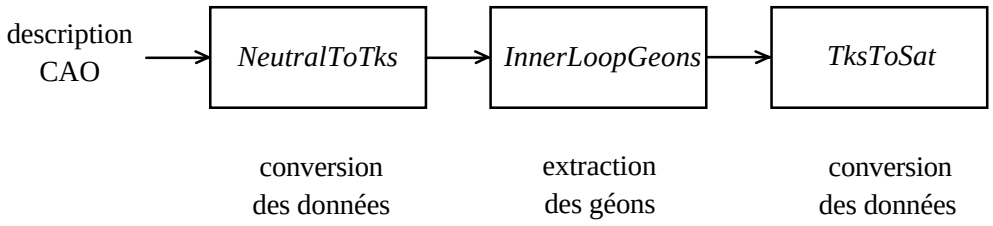


Figure 4.1 : Enchaînement des programmes.

4.2 Conversion des données CAO (*NeutralToTks*)

Tel qu'illustré à la figure précédente, nous avons choisi de distinguer le programme d'acquisition de données du programme d'analyse de données. Outre que cette séparation du code source soit assez naturelle, elle offre aussi plus de souplesse quant au type de format de fichier fourni en entrée. En effet, bien que nous utilisons actuellement un certain format comme source de données CAO, le découpage proposé permettrait de lire d'autre types de format de fichier à moindre coût : il suffirait alors d'écrire un programme de conversion de données spécifique au format de fichier envisagé, plutôt que de revoir tout le code source.

4.2.1 Choix de la représentation d'entrée

La toute première étape de ce projet consistait à choisir un format de fichier pour transférer la description CAO du modélisateur vers notre projet. Parmi les caractéristiques désirables d'un format de fichier CAO pour la reconnaissance volumétrique, nous mentionnerons les suivantes :

- La valeur sémantique des primitives, qui doit être aussi élevée que possible. Ce critère tend à exclure par exemple les représentations par voxels, unités aussi peu signifiantes en 3D que les pixels dans une image 2D. Autre exemple, du côté des représentations Brep : le format de fichier STL, constitué uniquement de primitives planes. Bien que ce format de fichier diminue notoirement la complexité algorithmique par la simplicité de ses primitives, il souffre aussi d'une faible valeur sémantique lorsque l'objet est non-polyédrique : par exemple, une seule surface cylindrique est modélisée en STL par plusieurs dizaines ou centaines de micro-facettes. Ce morcellement de l'information originelle irait tout à fait à l'encontre notre objectif de reconnaissance de forme.

- La disponibilité industrielle de la représentation CAO. L'objectif de ce projet étant résolument orienté vers une éventuelle application industrielle, il était effectivement impérieux que le format de fichier soit accessible en dehors des seuls laboratoires de recherche. Compte tenu des formats CAO communément utilisés en industrie, ce critère restreignait donc le choix à l'un ou l'autre des membres de la famille des Brep.

- La facilité de recouvrement des entités, qui suppose la présence de primitives géométriques usuelles, codifiées simplement. Un format de fichier qui n'utiliserait par exemple que des surfaces sculptées serait nettement défavorisé par ce critère, puisque ce type de primitive masque fortement la nature des surfaces en cause.

- La possibilité d'extensions futures au domaine de modélisation. Les pièces expérimentales actuelles sont constituées uniquement de surfaces planes, cylindriques et coniques, ce qui ne requiert qu'un nombre limité de primitives. Cependant, il serait manifestement déplorable d'avoir à utiliser un autre format de fichier par la suite, advenant le besoin de modéliser des entités plus complexes.

Ces différentes considérations nous ont incité à choisir le format *neutre*, écrit par le modélisateur ProEngineer. Ce format de fichier ne répond à aucun standard, et émane de la volonté d'offrir à l'usager une description de l'objet dans un langage relativement accessible.

On doit remarquer que plusieurs modélisateurs produisent des fichiers dits neutres pour satisfaire aux besoins d'applications comme la nôtre ; la structure de ces fichiers demeure cependant spécifique à chaque modélisateur. En général, ces différents formats neutres seront écrits en ASCII pour faciliter la lecture du fichier, et utiliseront une codification réduite au minimum afin de simplifier l'interprétation des données brutes. Le type de fichier neutre que nous traitons ne fait pas exception à cette règle.

Nous ne désirons pas ici entrer dans le détail du format neutre ; il suffira pour l'instant de savoir qu'il s'agit d'une description Brep utilisant une forme énumérative F(E), où les entités sont décrites paramétriquement. Les éléments les plus critiques de ce format de fichier seront mentionnés par la suite, selon les besoins. Notre lecteur pourra consulter l'annexe B pour une description détaillée de ce format de fichier.

Tel que nous le verrons plus loin, la forme paramétrique présente certains avantages pour les calculs, surtout parce qu'elle constitue une représentation en deux dimensions d'une entité 3D. Le format neutre souffre néanmoins d'une certaine faiblesse descriptive, surtout quant au nombre de primitives d'arêtes. À noter que le format neutre ne supporte par ailleurs aucune primitive volumétrique, ce qui exclut toute écriture de la description géon finale sous ce format.

4.2.2 Processus de conversion des données

Très sommairement, on peut distinguer trois grandes fonctions au sein de *NeutralToTks* :

1. La discrimination des données. Le fichier neutre présente une information abondante sur l'objet, mais dont une partie seulement nous interesse. Il appartient donc au programme de faire le tri parmi cette masse de données.

2. La traduction des données. Outre qu'elle doive faciliter l'analyse ultérieure, la structure de données produite en sortie doit aussi s'accomoder des schémas de représentation disponibles dans l'environnement KBV ; la tâche *NeutralToTks* doit donc traduire les données, du langage du modélisateur vers le langage de KBV. À cet égard, la tâche génère des *éléments symboliques*, qui ne sont rien d'autre que des structures au sens du C, regroupant ensemble toutes les informations relatives à une entité. Cette façon d'agglomérer l'information convient parfaitement à nos besoins,

d'autant plus qu'il existe déjà dans l'environnement KBV de nombreuses fonctions pour manipuler les éléments symboliques.

3. L'inférence sur les données. Il ne s'agit pas ici de rechercher les géons dans la description fournie, mais bien de créer un jeu de données qui soit *complet*. Nous avons mentionné en effet que le format neutre constitue une énumération F(E) des entités de l'objet, et nous avons évoqué au chapitre 1 les inconvénients relatifs à cette forme énumérative. Le programme devra donc identifier correctement les boucles et sommets de l'objet, afin de produire une forme F(L(E(V))), plus complète au sens de la reconnaissance de forme. La figure 4.2 résume la transformation à appliquer sur les données.

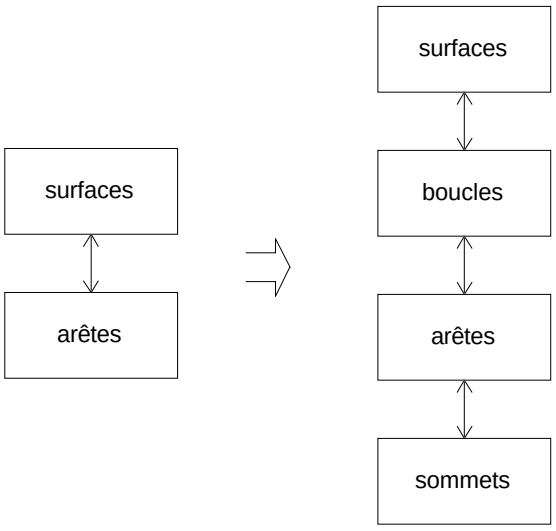


Figure 4.2 : Transformation des données.

Tel que l'illustre la figure précédente, la forme énumérative F(L(E(V))) délimite l'extension spatiale des surfaces par le biais des boucles, qui ont maintenant une identité distincte. Le résultat de cette transformation s'apparente à la structure de données dite de demie-arête (*half-edge data structure*), couramment utilisée dans le domaine. Puisque

chaque arête appartient simultanément à deux surfaces, une arête physique peut effectivement être représentée par deux entités logiques, chacune orientée en sens inverse de l'autre.

Notre structure de données diffère quelque peu de ce schéma, dans la mesure où nous associons une seule arête logique à chaque arête physique, pour alléger un peu la représentation. Autre différence avec la structure de demie-arête, l'absence de pointeurs latéraux, qui indiquent les entités suivante et précédente d'une liste. Bien que le parcours séquentiel des entités soit souvent requis, nous n'avons jamais ressenti un besoin criant de pointeurs latéraux, essentiellement parce que l'analyse s'effectue du haut vers le bas, c'est-à-dire à partir d'entités qui indiquent déjà la séquence des entités inférieures[8].

Tel que nous le verrons plus loin, une certaine dose d'analyse est requise par *NeutralToTks* pour reconstituer correctement la structure de données F(L(E(V))) à partir de l'information disponible dans le fichier neutre.

4.2.3 Paramètres d'entrée/sortie

La tâche *NeutralToTks* reçoit en entrée le fichier neutre, et génère en sortie quatre fichiers d'éléments symboliques, correspondant aux surfaces, aux boucles, aux arêtes et aux sommets de l'objet. L'objectif général du module CAO→géons consiste évidemment à produire un cinquième ensemble symbolique, celui des géons.

Concrètement, *NeutralToTks* requiert six paramètres spécifiques :

1. Le nom du fichier neutre à lire ;

---

[8] En fait, la tâche suivante ajoutera pour les besoins de son analyse des pointeurs latéraux entre les surfaces, puisqu'il n'existe initialement aucune entité supérieure aux surfaces.

2. Le nom à donner à chacun des fichiers de sortie. Par défaut, la tâche ajoutera au nom du fichier neutre l'extension _S, _L, _E ou _X, pour identifier respectivement les fichiers de surfaces, de boucles, d'arêtes et de sommets ;
3. Le seuil de remise à zéro des petits éléments.

Nous reviendrons plus loin sur la signification du dernier paramètre. Chaque élément symbolique résume l'information géométrique essentielle pour décrire l'entité qu'il représente ; les formes mathématiques utilisées pour représenter les différentes entités sont expliquées à l'annexe C. Examinons brièvement la forme des éléments symboliques produits en sortie.

1. Éléments symboliques de surface

Chaque entité de surface est caractérisée par huit attributs ; il s'agit cependant d'une forme générale, puisque certains attributs n'ont de sens que pour un type particulier de surface. Le nombre et la nature de ces attributs sont orientés en fonction des capacités actuelles d'analyse de *InnerLoopGeons*. Les attributs de surface sont :

1. Type..... Surface plane, cylindrique ou conique, qui sont les seuls types de surface admis actuellement.
2. Origine..... Localisation (x,y,z) du repère local de la surface, selon les modèles de paramétrisation expliqués à l'annexe C.
3. Normale ..... Vecteur unitaire normal à la surface. Défini seulement pour un plan.
4. Boucles ..... Pointeurs vers les boucles de la surface. Il aurait été restrictif de mémoriser les boucles directement dans le fichier des surfaces, puisque le nombre de boucles associées à chaque surface est imprévisible. Par convention, la première boucle mentionnée est la boucle externe de la surface.
5. Convexité..... Drapeau qui indique si le côté matériel d'une surface cylindrique ou conique est situé vers son centre de courbure.

6. Axe ..... Vecteur unitaire qui indique l'orientation de l'axe d'une surface cylindrique ou conique.
7. Rayon..... Pour une surface cylindrique.
8. Demi-angle..... Angle d'ouverture d'une surface conique.

2. Éléments symboliques de boucle

Chaque boucle mentionnée dans cet ensemble possède trois attributs :

1. Type..... Externe ou interne. Bien que le type de boucle soit aussi disponible parmi les attributs de la surface parente, il est plus rapide pour certaines opérations de disposer localement de cette information.
2. Arêtes..... Pointeurs vers les arêtes de la boucle.
3. Parent..... Pointeur vers la surface parente.

3. Éléments symboliques d'arête

Chaque arête dispose de onze attributs ; comme pour les surfaces, certains attributs sont spécifiques à des types particuliers d'arête :

1. Type..... Ligne droite, arc de cercle ou B-spline, qui sont d'ailleurs les seules primitives d'arête disponibles dans le format neutre. Bien que nous admettons en principe les B-spline parmi les données, nous ne disposons à date d'aucun outil d'analyse à leur égard. Ceci explique pourquoi les noeuds et points de contrôle du spline ne sont pas mémorisés.
2. SurfaceList..... Pointeurs vers les surfaces voisines de l'arête.
3. StartPoint ..... Pointeur vers le sommet de départ de l'arête.
4. EndPoint ..... Pointeur vers le sommet d'arrivée de l'arête.
5. Direction ..... Drapeaux indiquant le sens de parcours de l'arête relativement à chacune des surfaces voisines.

6. Centre ..... Localisation (x,y,z) du centre d'un arc de cercle.
7. Rayon..... Pour un arc de cercle.
8. Angle ..... Angle d'ouverture d'un arc de cercle.
9. Axe ..... Vecteur unitaire qui indique l'orientation de l'axe d'un arc de cercle.
10. Vecteur1 ..... Vecteur unitaire parallèle au plan porteur d'un arc de cercle.
11. Vecteur2 ..... Vecteur unitaire perpendiculaire à Vecteur1, et parallèle au plan porteur de l'arc de cercle.

4. Éléments symboliques de sommet

Chaque sommet ne possède que deux attributs, à savoir :

1. Position ..... Localisation (x,y,z) du sommet.
2. EdgeList..... Pointeurs vers les arêtes qui se joignent à cet endroit.

Pour résumer l'ensemble des liens entre les entités, nous illustrons à la figure 4.3 les différents pointeurs impliqués dans notre structure de données.

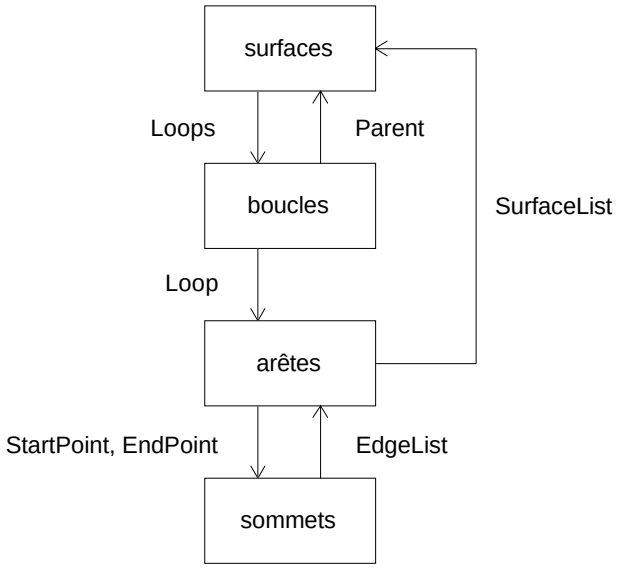


Figure 4.3 : Pointeurs entre les entités.

Pour les éléments symboliques de surfaces et d'arêtes, l'index de l'entité correspond simplement à son étiquette telle qu'elle apparaît dans le fichier neutre ; les boucles et les sommets sont quant à eux numérotés séquentiellement par *NeutralToTks*. Chaque ensemble symbolique maintient parmi ses propriétés le nom du fichier neutre d'origine, le nombre d'entités qu'il contient, ainsi que les équivalences numériques de certains types d'attributs.

4.2.4 Algorithmes

Dans le cas le plus général, la tâche procède à l'analyse des données en trois étapes :

1. Lecture des données brutes.
2. Analyse des boucles multiples.
3. Analyse des sommets.

Nous ne nous attarderons pas sur la première étape, qui ne présente aucune difficulté particulière. Il s'agit simplement pour le programme de passer à travers les sections "Surfaces" et "Arêtes" du fichier neutre, et de mémoriser les données immédiatement disponibles. À la fin de ce premier passage, *NeutralToTks* aura initialisé les ensembles symboliques de surfaces, de boucles et d'arêtes ; restera encore à identifier les sommets, et à revoir au besoin les surfaces et les boucles.

4.2.4.1 Analyse des boucles multiples

Si une surface contient plusieurs boucles, rien dans le fichier neutre n'indique laquelle est externe et lesquelles sont internes ; cette information est pourtant essentielle pour nous, puisqu'elle est le fondement même de notre stratégie d'extraction de géons.

Le cas échéant, la tâche *NeutralToTks* doit donc comparer l'emplacement de toutes les boucles appartenant à une même surface pour identifier les boucles externes et internes. Le problème n'a rien d'insoluble, puisqu'une boucle interne est toujours située à l'intérieur d'une boucle externe dans leur domaine de définition commun, c'est-à-dire dans le domaine de la surface.

Il faudra cependant considérer un autre difficulté lors de l'analyse des boucles multiples, à savoir qu'une seule et même surface peut être définie par plus d'une boucle externe, selon la méthode de construction choisie par l'usager. L'objet de la figure 4.4 (a) par exemple a été construit en une seule opération, par extrusion de sa face avant le long d'un axe droit ; la présence des deux surfaces supérieures sera alors clairement exprimée dans la description CAO.

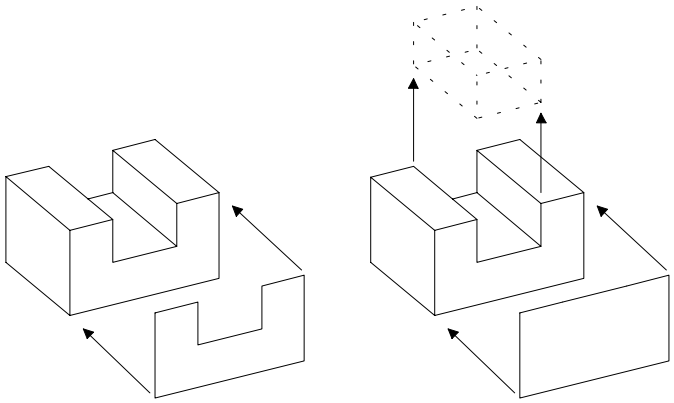


Figure 4.4 : Boucles externes multiples.

Cependant, le même objet pourrait aussi être modélisé en deux opérations, tel qu'en 4.4 (b) : en définissant d'abord un parallélépipède complet, auquel on retranche par la suite une rainure horizontale. Le modélisateur assignera alors une seule entité logique pour la surface supérieure, avec un domaine morcelé en deux régions disjointes.

Pour ce dernier cas, le fichier neutre indiquera donc la présence d'une seule surface, définie par deux boucles. Cette description est mathématiquement exacte, puisque la paramétrisation d'une des surfaces planes s'applique aussi à l'autre.

L'éventuelle multiplicité de boucles externes n'est donc pas une erreur d'écriture du fichier : les données sont simplement mémorisées sous une forme suffisante pour la modélisation, alors que notre besoin de distinguer chaque surface physique découle plutôt d'un objectif de reconnaissance de forme. On notera au demeurant que la multiplicité de boucles externes ainsi que l'absence de marqueur pour distinguer les boucles externes et internes ne sont pas uniques au format neutre.

Bref, en cas de boucles multiples, il faut prévoir la possibilité qu'il y ait non pas une seule, mais plusieurs boucles externes, et éventuellement aucune ou plusieurs boucles internes. Ceci oblige donc à analyser toutes les boucles deux à deux, dans l'éventualité où une surface contiendrait plusieurs boucles externes, se partageant plusieurs boucles internes.

Une question mérite d'être soulevée ici : est-il vraiment utile de distinguer les surfaces définies par plusieurs boucles externes ? En effet, si la rainure de l'objet de la figure 4.4 correspond à un géon négatif, la tâche *InnerLoopGeons* devrait ensuite réunir les deux surfaces supérieures lors de la complétion de modèle[9]. À ce titre, si la description CAO indique déjà que ces deux surfaces que nous voulons séparer appartiennent au même plan, ne vaudrait-il pas mieux préserver cette information ? Pour ce cas, oui ; nous avons cependant confirmé que des parties séparées d'une même surface ne sont pas toujours destinées à faire partie du même géon. Nous préférons donc que *InnerLoopGeons* amorce son raisonnement à partir de certitudes plutôt que de risquer des cas d'exception.

---

[9] L'hypothèse d'un géon négatif à la place de la rainure est en fait contradictoire avec la définition fonctionnelle des géons de ce projet. Ceci n'invalide cependant pas la pertinence de la question initale.

Bien que l'analyse des boucles multiples pourrait s'effectuer directement dans l'espace 3D, ceci deviendrait problématique pour les boucles d'une surface non-plane. La figure 4.5 illustre le cas d'une surface cylindrique à trois boucles : il serait effectivement laborieux de coder une procédure qui puisse conclure que la boucle L2 est comprise à l'intérieur de L1, et que L3 est à l'extérieur de L2. Une analyse fondée par exemple sur la position 3D des sommets de chaque boucle serait particulièrement inefficace pour conclure quoi que ce soit. On pourrait aussi songer à comparer les parallélépipèdes extrêmes (*bounding box*) de chaque boucle ; encore ici, il y aurait problème avec les boucles L2 et L3, puisque le parallélépipède de l'un serait contenu dans l'autre.

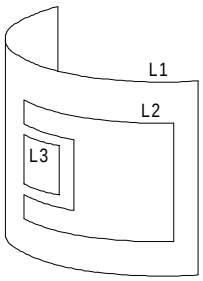


Figure 4.5 : Boucles d'une surface cylindrique.

Bref, nous croyons qu'il est autrement plus simple de chercher à résoudre le problème dans un espace 2D. Une approche possible serait de projeter toutes les boucles sur un certain plan, dont la position serait à déterminer ; l'espace paramétrique des surfaces permet cependant d'éviter ce calcul. En effet, une surface est toujours paramétrée en fonction de deux variables : le passage de l'espace géométrique (x,y,z) à l'espace paramétrique (u,v) équivaut à la transformation d'une entité 3D en une entité 2D équivalente[10]. Outre que cet espace nous évite le calcul d'un plan de projection, il nous

---

[10] La paramétrisation n'est rien d'autre qu'un changement de référentiel. Par exemple, une boucle planaire est paramétrée dans un système de coordonnées rectangulaires centré sur le plan porteur, tandis que les boucles cylindriques et coniques sont exprimées dans un référentiel cylindrique centré sur le cylindre ou le cône porteur. À cet égard, la paramétrisation peut changer la forme de la boucle, mais elle préserve cependant la position relative de deux boucles, condition nécessaire à notre analyse.

permet aussi de bénéficier de la disponibilité des paramètres (u,v) des arêtes dans le fichier neutre.

Pour fins d'analyse, les boucles multiples sont approximées dans l'espace paramétrique par un contour fermé d'arêtes droites. La polygonisation de primitives courbes est en effet une technique éprouvée pour réduire la complexité géométrique, et constitue d'autre part une interprétation suffisante des paramètres (u,v) mentionnés dans le fichier neutre.

Reste alors à comparer toutes les boucles entre elles, et à vérifier lesquelles sont incluses dans quelles autres. Il est d'ailleurs utile de remarquer que si un sommet d'une boucle L1 est situé à l'intérieur d'une boucle L2, alors tous les sommets de L1 doivent aussi être à l'intérieur de L2, tel qu'illustré à la figure 4.6. Ce principe permet d'éviter de coûteuses vérifications, puisqu'il suffit de situer un seul des sommets d'une boucle par rapport à l'autre boucle pour conclure sur la position respective des deux boucles.

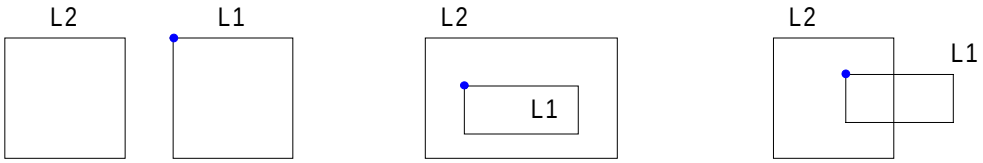
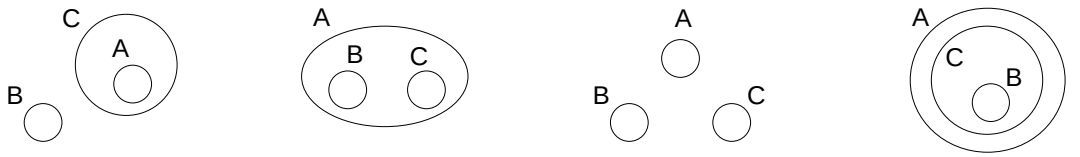


Figure 4.6 : Contrainte d'inclusion ou d'exclusion complète des boucles. Le cas illustré à l'extrême-droite est impossible, puisque l'objet correspondant est incohérent au sens d'Euler.

Notre algorithme procède par vérification 2 à 2 de toutes les boucles multiples d'une même surface, et les résultats sont mémorisés dans ce que nous avons appelé la *matrice d'inclusion*. La figure 4.7 illustre quelques cas de boucles multiples, et de leurs matrices d'inclusion respectives. Pour chacun de ces cas, l'algorithme vérifierait de trois à six possibilités d'inclusion : par exemple, si A est inclus dans C (fig. 4.7 (a)), le test est concluant ; par contre, si A n'est pas inclus dans C, il faudra nécessairement vérifier la

relation inverse, puisque C peut être inclus dans A ou non (fig. 4.7, respectivement (b) et (c)).

L'algorithme termine son analyse par l'étude du nombre de 1 dans la matrice d'inclusion. Par exemple, une colonne qui contient un nombre pair de 1 (y compris aucun 1) indique une boucle externe, alors qu'une colonne qui contient un nombre impair de 1 indique une boucle interne. Remarquer d'ailleurs le cas 4.7 (d), où B est inclus dans C, qui est inclus dans A : bien que C soit une boucle interne, la double inclusion fait de B une boucle externe, ce qu'indique effectivement la règle de parité des 1 sur une colonne.



	A	B	C
A	0	0	0
B	0	0	0
C	1	0	0

(a)

	A	B	C
A	0	1	1
B	0	0	0
C	0	0	0

(b)

	A	B	C
A	0	0	0
B	0	0	0
C	0	0	0

(c)

	A	B	C
A	0	1	1
B	0	0	0
C	0	1	0

(d)

Figure 4.7 : Matrices d'inclusion.

La tâche *NeutralToTks* devra créer et initialiser de nouvelles entités logiques de surface pour chaque boucle externe détectée. Lorsqu'il y a autant d'entités de surfaces que de boucles externes, le programme n'a plus qu'à répartir correctement les boucles internes parmi ces surfaces. Il procède alors par recoupement des lignes et colonnes de la matrice d'inclusion, pour bien départager l'assignation des boucles à inclusion multiple.

4.2.4.2 Analyse des sommets

Tel que mentionné auparavant, un objectif principal de *NeutralToTks* est de convertir la description CAO sous forme F(L(E(V))). Puisque les données brutes sont exprimées sous forme F(E), la tâche doit donc recomposer elle-même la liste des sommets de l'objet à partir de l'information disponible. Ceci suppose deux opérations distinctes, à savoir :

1. L'identification des sommets. Il s'agit dans un premier temps de déterminer le nombre de jonctions d'arêtes, ainsi que l'identité des arêtes qui se joignent à chaque sommet. Noter que la formule d'Euler ne permettrait pas de calculer le nombre de sommets, puisque le nombre de trous dans l'objet est encore inconnu.

2. La localisation des sommets. Puisque le format neutre exprime les données exclusivement sous forme paramétrique, il faut aussi calculer l'emplacement des extrémités d'arêtes. En effet, la position d'un point paramétrique est connue uniquement dans le référentiel de la surface d'appartenance ; les données brutes n'expriment donc qu'indirectement l'emplacement des jonctions d'arêtes. De façon plus générale, la localisation des sommets consiste à exprimer la position de tous les sommets par rapport à une seule référence, à savoir le repère universel de l'objet.

L'identification des sommets ne pose pas de grande difficulté au niveau algorithmique, d'autant plus que les boucles de l'objet sont maintenant connues. En effet, tout trajet fermé dans le graphe relationnel des arêtes identifie un sommet distinct, alors que les noeuds traversés indiquent les arêtes qui se rencontrent à ce sommet[11]. La figure 4.8 illustre cette technique d'identification des sommets, mise en oeuvre à partir de la liste des boucles externes.

---

[11] Ceci n'est vrai cependant que si l'objet est un solide cohérent au sens d'Euler, ce que nous prenons pour acquis..

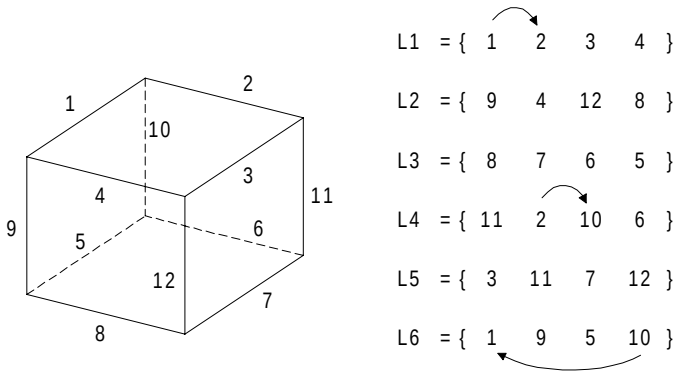


Figure 4.8 : Algorithme d'identification des sommets.

De façon très succincte, cet algorithme se résume à :

1. Choisir une arête X de départ ;
2. Déterminer l'arête Y suivante dans la boucle ;
3. Est-ce qu'il s'agit de l'arête de départ ?
   oui : sommet identifié. Reprendre à l'étape 1.
   non : rechercher l'autre occurrence de l'arête Y dans les autres boucles ;
4. Continuer à l'étape 2

Dans l'exemple de la figure 4.8, X = 1 ; à la fin de la première itération, le programme aura donc déterminé que les arêtes 1, 2, et 10 se joignent en un certain sommet, dont les coordonnées restent à déterminer. Il n'est d'ailleurs pas nécessaire d'appliquer cette procédure pour toutes les arêtes de toutes les boucles, puisqu'une même arête ne peut jamais être incidente à plus de deux sommets.

Cet algorithme emprunté aux techniques de graphe présente un intérêt évident, puisqu'il permet d'identifier toutes les arêtes incidentes à un sommet sans tenir compte d'aucune coordonnée géométrique. La recherche des sommets par comparaison directe des coordonnées (x,y,z) des arêtes serait très coûteuse, puisqu'un seul objet peut présenter plusieurs dizaines, voire centaines, de sommets. D'autre part, l'examen d'une

seule boucle à la fois ne permettrait pas d'identifier des sommets toujours distincts des précédents, ni de connaître toutes les arêtes incidentes au sommet.

Une fois tous les sommets identifiés, reste donc à calculer leur position (x,y,z) et à vérifier s'ils correspondent au point de départ ou au point d'arrivée des arêtes incidentes. Cependant, la situation se complique un peu en raison des erreurs d'arrondi, qui émanent de deux sources distinctes :

1. Des erreurs d'arrondi sont déjà observables dans le fichier neutre : autrement dit, les données brutes ne sont pas parfaitement fiables. Ces erreurs apparaissent à plusieurs endroits de la description : origine des repères locaux, direction des axes, etc. Pour des entités courbes par exemple, le modélisateur doit nécessairement manipuler des fonctions trigonométriques, dont les problèmes de convergence provoquent un déplacement minime mais réel des points[12].

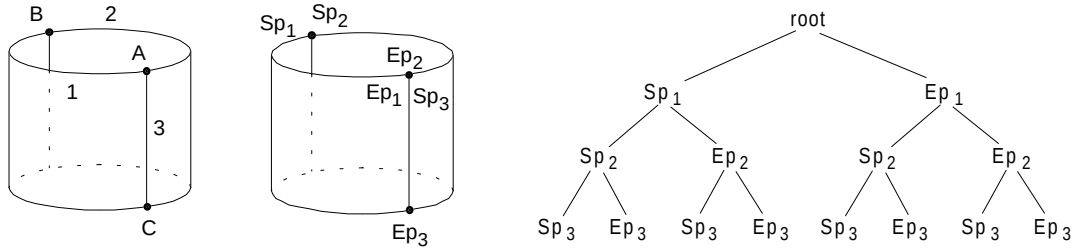
   Le paramètre d'entrée *TinyElements* comble très partiellement cette lacune des données initiales, en forçant une remise à zéro des valeurs plus petites qu'un certain seuil. Ce paramètre agit sélectivement, pour ne remettre à zéro que les grandeurs sujettes aux erreurs d'arrondi.

2. Tel que mentionné précédemment, la tâche *NeutralToTks* doit calculer elle-même les coordonnées (x,y,z) de début et de fin d'arête en fonction des valeurs (u,v) des paramètres, engendrant ainsi d'autres erreurs d'arrondi. L'erreur de localisation (Δx, Δy, Δz) s'en trouve multipliée, puisque les valeurs (u,v) utilisées pour le calcul peuvent être déjà fausses, suite aux erreurs d'arrondi du modélisateur. Si l'erreur de calcul sur x, y et z demeure généralement faible (de l'ordre de 10^{-10}), nous avons cependant observé qu'elle entraîne parfois des erreurs aussi élevées que 10^{-3}.

---

[12] Les erreurs d'arrondi constituent un problème pour tous les modélisateurs. À ce titre, le format neutre ne devrait pas être considéré moins précis que n'importe quel autre format de fichier CAO.

Bref, la position du sommet n'est pas connue avec certitude, puisque chaque extrémité d'arête indique un emplacement légèrement différent ; d'autre part, chaque arête comporte *deux* extrémités, dont les coordonnées devront nécessairement être comparées bien qu'elles soient imprécises. Pour compléter le tableau, reste à évoquer le problème des arêtes en boucle, illustré à la figure 4.9.



(a) (b) (c)

Figure 4.9 : Arêtes en boucle. (a) Arêtes et sommets identifiés. (b) Coordonnées connues, avec Sp : StartPoint et Ep : EndPoint. (c) Arborescence d'appariement.

Pour ce cas particulier, mais très fréquent parmi nos pièces expérimentales, le programme doit tenir compte qu'il existe éventuellement deux solutions à l'appariement des extrémités d'arêtes. Pour l'objet de la figure 4.9 (a), considérons par exemple que le programme connaît l'existence d'un sommet A, lieu de jonction des arêtes 1, 2 et 3 ; la position de A est inconnue, et ne pourra être déterminée qu'en comparant les coordonnées des extrémités d'arêtes. Si le programme commence d'abord par comparer les extrémités des arêtes 1 et 2, il obtient alors deux solutions : il doit donc nécessairement comparer les extrémités de *toutes* les arêtes incidentes pour parvenir à une solution unique. Il faut donc d'explorer l'arborescence complète des possibilités, tel qu'illustré en 4.9 (b), afin d'éviter toute confusion de sommets.

La solution consiste évidemment à tolérer une certaine déviation lors de l'appariement, et à essayer de mémoriser ensuite la meilleure valeur possible des coordonnées

du sommet. Toute cette minutie est requise pour éviter à *InnerLoopGeons* de parvenir à de fausses conclusions causées par des erreurs sur les données. Notre algorithme prévoit donc trois étapes principales :

1. Calcul de la tolérance de localisation. La méthode la plus simple consiste à tolérer une déviation égale à la moitié de la plus petite distance entre toutes les extrémités d'arêtes incidentes :

$$\text{tolérance} = \frac{1}{2} \min \left\| < Sp_i, Ep_i > \right\|$$

   La distance entre les extrémités d'arêtes est toujours prise en ligne droite, même pour une arête courbe, puisqu'il s'agit de la plus petite distance entre ces points, ce qui engendre donc la tolérance la plus faible possible. Remarquons d'autre part que la présence d'arêtes étrangères à l'intérieur de la zone de tolérance serait sans effet sur le processus d'appariement, puisque l'espace d'analyse est restreint aux arêtes incidentes.

2. Identification des extrémités d'arêtes. Une fonction récursive explore l'arborescence des possibilités, et élimine ainsi les coordonnées qui n'appartiennent pas au sommet étudié.

3. Calcul de la position du sommet. Nous utilisons la valeur moyenne des coordonnées comme position du sommet.

4.3 Extraction de géons sur boucles internes (*InnerLoopGeons*)

4.3.1 Présomptions sur le domaine d'entrée

Le domaine d'entrée de ce projet se limite aux dix pièces de la figure 0.3 (p. 6). Nos algorithmes présument donc d'un certain nombre de conditions géométriques, définies à partir des pièces expérimentales :

Contraintes sur les pièces :

1. Les pièces sont formées uniquement d'un assemblage de cylindres généralisés. Dans le cadre d'une approche géon, cette condition apparaît effectivement comme un minimum absolu, puisqu'elle permet l'existence d'au moins une solution valide pour tout objet soumis au module.

2. Étant donné la grande variété d'entités géométriques possibles, les programmes implémentés à date admettent uniquement comme arêtes les lignes et les arcs de cercle, et comme surfaces les plans, les cylindres et les cônes. Cette condition exclut donc la présence d'arêtes splines ou de surfaces NURBS dans la description CAO. Le projet en étant à ses débuts, il importe en effet de pouvoir reconnaître des entités simples avant d'entreprendre l'analyse de cas plus complexes.

3. La pièce ne devrait pas présenter de transitions d'arêtes, donc de chanfreins, congés, ou arrondis. Ceci vise à éliminer les entités qui ne contribuent pas significativement à l'identification de l'objet, et qui compliqueraient le développement des algorithmes. Dans leur forme actuelle, nos algorithmes identifieraient possiblement une transition d'arête comme un géon en soi.

Contraintes sur les géons :

4. Les cylindres généralisés doivent être homogènes, c'est-à-dire maintenir un profil générateur de forme constante au cours du balayage. Cette contrainte constitue la norme dans la presque totalité des travaux puisque les corps de forme variable sont très peu fréquents. On doit noter d'autre part qu'il serait impossible de modéliser un cylindre généralisé non-homogène avec les primitives géométriques conventionnelles ; la contrainte d'homogénéité est donc une corollaire à la condition # 2, qui proscrit les surfaces sculptées.

5. La surface porteuse du profil générateur doit être perpendiculaire à l'axe de balayage. Bien que cette contrainte exclue la reconnaissance de certains volumes tronqués en oblique, tel qu'illustré à la figure 4.10 (a), elle évite aussi d'avoir plusieurs solutions pour un même géon : en témoigne l'objet en 4.10 (b), dont le profil générateur pourrait tout aussi bien être trapézoïdal que rectangulaire ; chaque interprétation mènerait alors à une conclusion différente quant à l'identité du géon. Avec cette contrainte de perpendicularité, notre programme n'accepterait qu'une seule solution pour l'objet de la figure 4.10 (b), à savoir qu'il s'agit du géon # 1.

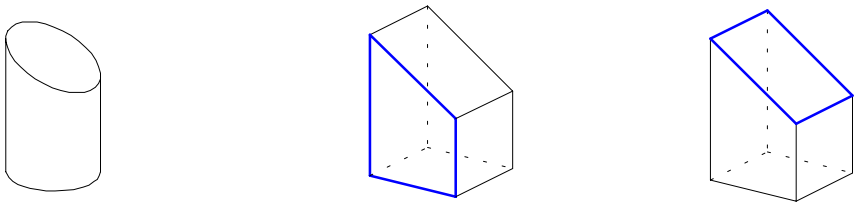


Figure 4.10 : Contrainte de perpendicularité.

6. La surface porteuse du profil générateur doit être plane, avec pour seule exception les cônes et les pyramides, dont l'un des profils générateurs est un apex. En fait, il s'agit plus d'une contrainte de géonisation que de modélisation. Compte tenu de la contrainte # 5, les extrémités des géons doivent donc former des plans parallèles.

7. L'axe du cylindre généralisé doit être droit. Puisqu'aucune pièce expérimentale ne présente de géon incurvé, il nous était loisible d'admettre cette contrainte qui diminue notablement la complexité de calcul des attributs. Dans l'hypothèse où on chercherait à élargir le domaine d'entrée du projet, il s'agirait cependant de la première contrainte à lever, puisqu'elle exclut d'emblée la moitié des géons possibles.

On notera que la pièce 7 ne respecte pas tout à fait la contrainte # 2, certaines de ses arêtes étant modélisées par des splines. Ceci demeure sans grande importance pour notre projet, puisque cette pièce ne possède aucune boucle interne et qu'elle est donc hors de portée de *InnerLoopGeons*.

4.3.2 Paramètres d'entrée/sortie

La tâche reçoit en entrée les quatre fichiers produits par *NeutralToTks*, et qui énumèrent l'ensemble des surfaces, boucles, arêtes et sommets de l'objet. Bien que *InnerLoopGeons* devrait écrire en principe un unique fichier de sortie, celui des géons, cette tâche génère en fait 10 fichiers distincts ; ceci mérite explication.

Tel que mentionné auparavant, la tâche *InnerLoopGeons* doit préserver toute l'information géométrique pour les programmes ultérieurs[13] ; on parle ici du module d'affichage, permettant la visualisation des géons, et du programme de calcul de connectivité, qui ne devrait s'amorcer qu'après identification de tous les géons de l'objet[14]. À ce titre, un géon identifié est écrit dans l'ensemble symbolique des géons, alors que ses entités membres (surfaces, boucles, arêtes et sommets) sont sauvegardées séparément des entités non-reconnues.

Dans notre nomenclature, nous qualifions ces derniers fichiers de "ensembles de mémorisation", puisque leur but essentiel est de préserver l'information géométrique, tout en éliminant les entités reconnues de l'espace d'analyse. Il s'agit en l'occurrence des cinq fichiers à droite de la figure 4.11.

---

13 Ces programmes s'insèrent en aval de ceux illustrés à la fig. 4.1 (p. 92). Ils n'y apparaissent pas, l'intention de cette figure étant seulement de résumer le travail que nous avons effectué ; le lecteur pourra cependant référer à la fig. 6.4 (p. 157) pour une vue d'ensemble de tous les programmes impliqués dans le module CAO→géons.

14 Bien qu'il y aurait moyen de déterminer la connectivité entre les géons reconnus, notre programme aura plutôt tendance à identifier des géons non-connectés, d'où le peu d'intérêt à entreprendre immédiatement le calcul de connectivité.

D'autre part, la tâche *InnerLoopGeons* est incapable d'extraire tous les géons de l'objet, entre autre parce qu'elle n'applique qu'une seule méthode de segmentation ; ceci signifie donc qu'il restera encore des entités non-reconnues à la fin de l'analyse. La poursuite du projet implique que cette information soit passée intégralement à un autre programme, qui devra compléter l'extraction des géons à partir de cette description simplifiée de l'objet. Ceci ajoute donc quatre autres fichiers de sortie, qui correspondent aux "ensembles d'analyse".

Reste un dernier ensemble symbolique que *InnerLooopGeons* utilise comme espace de travail, celui des hypothèses de géon. Bien qu'il aurait pu être éliminé en fin

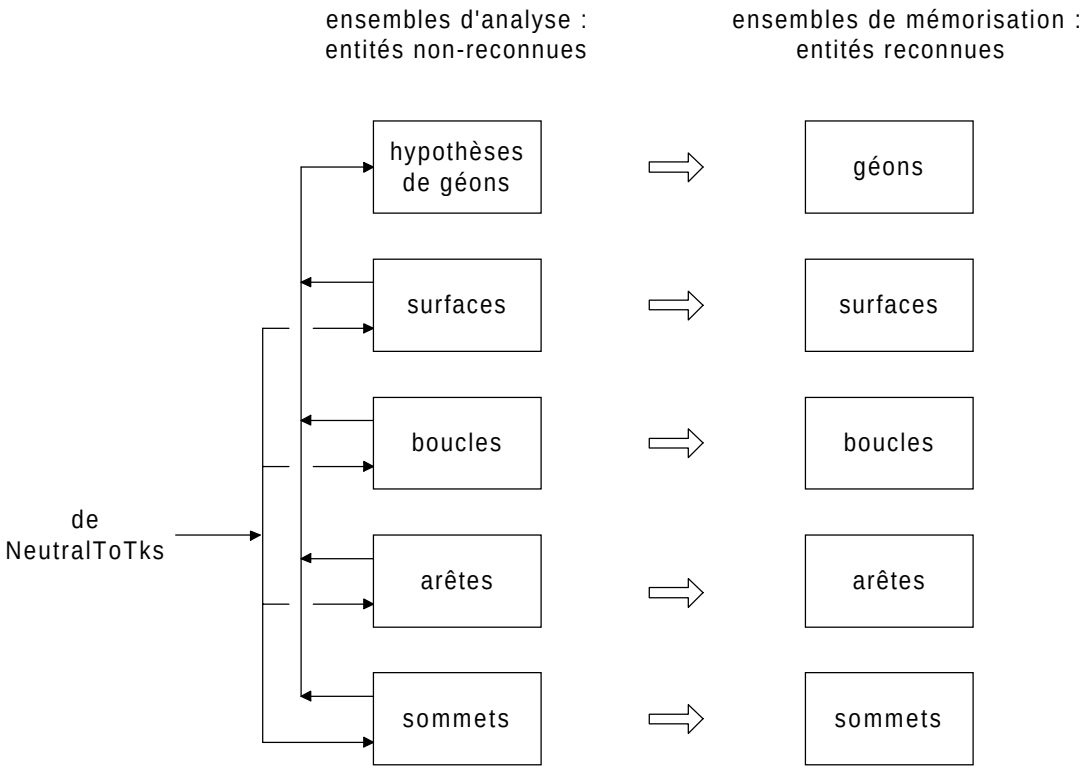


Figure 4.11 : Schéma de mémorisation de l'information. La tâche reçoit initialement 4 fichiers, qui servent à générer les hypothèses. Lorsqu'une hypothèse de géon est confirmée, le géon et ses descendants sont transférés vers l'espace de mémorisation.

de processus, nous avons choisi de passer ce fichier au programme qui suivra, lui évitant ainsi de ré-initialiser un espace de travail semblable. Ce fichier doit normalement être vide lorsque l'analyse se termine ; la présence d'hypothèses irrésolues dans ce fichier indiquerait alors un problème d'exécution de notre tâche.

On doit souligner que les ensembles d'analyse produits en sortie diffèrent sensiblement des fichiers d'entrée, bien qu'ils en émanent. Deux facteurs expliquent cette différence :

- Comme nous l'avons mentionné, les entités sont progressivement transférées vers les ensembles de mémorisation. À ce titre, les ensembles d'analyse contiennent une partie seulement de l'information présente dans la description CAO d'entrée ;

- Facteur plus important cependant, le fait qu'il y a une nécessaire évolution de la description à mesure que les géons sont reconnus. La phase de complétion de modèle implique en effet de remodeler des surfaces, d'ajouter ou fusionner des arêtes, de créer de nouvelles surfaces, etc. Il s'ajoute donc dans l'espace d'analyse de nouvelles entités, initialement absentes de la description CAO, alors que d'autres entités de la description d'origine y apparaissent maintenant sous une forme modifiée.

Il sera pertinent d'examiner la nature de l'information mémorisée pour chaque géon. Chaque géon est décrit par 10 attributs : cinq d'entre eux correspondent aux attributs qualitatifs, alors que les cinq autres indiquent les entités géométriques qui forment le géon.

<u>Attributs qualitatifs :</u>

1. AxisType..... Axe droit ou incurvé.
2. EdgeType..... Arêtes génératrices droites, courbes ou hybrides.
3. MaterialType..... Géon positif ou négatif.

4. SweepFunction..... Fonction de balayage constante ou variable.
5. Symmetry..... Symétrie réflective, réflective et rotationnelle, ou asymétrie.

Attributs géométriques :

6. StartingLoop ..... Liste des arêtes du profil générateur de départ. Compte tenu de la méthodologie utilisée, il s'agira généralement d'une boucle interne de l'objet.
7. EndingLoop ..... Liste des arêtes du profil générateur d'arrivée.
8. Surfaces ..... Liste de toutes les surfaces du géon.
9. Ribbon ..... Liste des surfaces latérales du géon.
10. EdgeLinks..... Liste des arêtes longitudinales du géon. Ce sont les arêtes du géon qui relient les deux profils générateurs.

Notre lecteur aura compris que les attributs qualitatifs identifient le géon, alors que les attributs géométriques seront utilisés pour la visualisation et le calcul de connectivité.

4.3.3 Algorithmes

4.3.3.1 Algorithme général

L'examen du programme principal sera une façon efficace de synthétiser le mécanisme d'extraction de géons. Nous éviterons de lister le code source complet, pour nous concentrer plutôt sur les aspects essentiels du programme. Les extraits de programme sont imprimés en `Courier New` dans le texte.

La toute première étape consiste bien sûr à lire les données provenant de *NeutralToTks*, et à initialiser les ensembles de mémorisation :

```
/*---------- Initialisation des données ----------*/

ReadTokensets () ;
InitRecognizedTokensets () ;
```

Avant d'entreprendre l'extraction des géons, la tâche calculera d'abord la convexité des arêtes, et identifiera les voisins de chacune des surfaces. Ces informations, absentes de la description CAO parce que superflues pour la modélisation, seront par contre très utiles à différentes étapes du processus d'analyse :

```
/*------------ Calculs préliminaires -------------*/

for (i = 0 ; i < EdgeList->NumEntries ; i++)
   ComputeEdgeConvexity (EdgeList->IndexList[i]) ;

for (i = 0 ; i < SurfList->NumEntries ; i++)
   ComputeSurfaceNeighbours (SurfList->IndexList[i]) ;
```

Nous aurons l'occasion de revenir sur l'algorithme de calcul de la convexité ; il suffira pour l'instant de savoir que la fonction vérifie si chaque arête est concave ou convexe, et qu'elle mémorise cette information dans un nouvel attribut, défini seulement pour les arêtes. La convexité des arêtes a d'ailleurs été abondamment exploitée dans les travaux antérieurs, puisqu'elle permet de déduire rapidement certaines caractéristiques géométriques des surfaces adjacentes.

On aura compris que le calcul du voisinage des surfaces consiste plus globalement à établir le graphe relationnel des surfaces. Bien que notre méthodologie ne repose pas sur l'analyse du graphe proprement dit, cette structure de données n'en demeure pas moins un outil simple et efficace pour explorer localement le modèle. On notera d'ailleurs que les graphes sont présents dans la presque totalité des travaux, même si la technique d'identification utilisée leur est totalement étrangère.

Il est assez aisé d'établir le graphe des surfaces puisqu'il suffit d'examiner toutes les arêtes d'une surface, et d'identifier à chaque fois la surface située de l'autre côté de

l'arête. On se rappelle que l'attribut SurfaceList indique directement quelles sont les surfaces adjacentes à une arête.

Commence ensuite le travail de segmentation et d'extraction des géons. Le programme détermine dans un premier temps la convexité des boucles internes, qui permettra de filtrer les cas plus difficiles :

```
/*------- Segmentation sur boucle interne --------*/

ComputeInnerLoopConvexity () ;
GeonizeOnInnerLoops (NewGeons) ;
```

La convexité d'une boucle est à l'image de la convexité de ses arêtes : concave ou convexe, si toutes ses arêtes sont concaves ou convexes, et hybride si certaines de ses arêtes sont concaves et d'autres convexes. Tel que nous le verrons au prochain chapitre, une boucle hybride nécessite un traitement plus élaboré, que nous n'avons d'ailleurs pas implémenté puisque aucune de nos pièces ne présente ce genre de situation. Dans la forme actuelle du programme, une boucle hybride provoquerait une fin prématurée d'exécution.

Tel que l'illustre l'extrait de code précédent, la segmentation a lieu immédiatement après le calcul de convexité des boucles. La fonction `GeonizeOnInnerLoops` examine successivement chaque boucle interne, et pose par hypothèse que cette boucle correspond au profil générateur d'un géon ; la phase ultérieure de validation devra confirmer qu'il existe bien un géon unique à cet endroit, effectivement engendré par le profil générateur présumé.

La phase d'extraction s'amorce dès qu'une boucle interne valide est détectée. Il s'agit en l'occurrence d'identifier les entités susceptibles d'appartenir au géon hypothétique : dans la mesure où le géon est essentiellement un volume, il semble naturel alors de rechercher les surfaces qui définissent ce volume.

La mécanique d'extraction applique une heuristique où interviennent des critères de voisinage de surfaces et de convexité des arêtes. Il faut remarquer au passage qu'il est difficile de procéder à cette étape autrement que par heuristique. En effet, la validation de l'hypothèse requiert un minimum d'entités pour fonder le raisonnement, le seul profil générateur étant très insuffisant pour conclure quoique ce soit ; d'autre part, la sélection des *bonnes* entités n'est possible que lorsque le géon est confirmé et identifié. Ce dilemme oblige à une approche par tentatives, le tout premier essai étant forcément guidé par l'empirisme.

Un résultat particulier de notre heuristique est que de nouvelles hypothèses de géon sont parfois générées durant l'extraction ; ainsi, notre programme pourrait trouver par exemple deux géons valides, là où n'existe pourtant qu'une seule boucle interne. La figure 4.12, qui illustre une partie de la pièce 6, permet de mettre en évidence ce résultat.

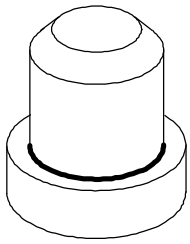


Figure 4.12 : Extraction de géons en enfilade.

Pour cet exemple simple, l'étape de segmentation localisera d'abord la boucle interne, illustrée d'un trait plus foncé, et le mécanisme d'extraction explorera le modèle à partir de cette boucle. Les deux surfaces qui forment le cylindre central seront immédiatement présumées faire partie du géon, puisqu'une de leurs arêtes appartient à la boucle interne. Parvenu en haut du cylindre, le programme concluera que le géon se termine apparemment à cet endroit, et terminera donc l'extraction.

En principe, le travail de segmentation/extraction devrait s'arrêter là, puisqu'il ne reste plus de boucles internes. Cependant, il est manifeste qu'un autre géon débute à l'endroit où se termine le premier : c'est justement ici que le programme tentera une nouvelle hypothèse de géon, pour laquelle l'heuristique confirmera ou non la pertinence d'enclencher une autre séquence d'extraction de surfaces. Lorsque la situation ne permet pas de croire à la présence immédiate d'un autre géon, aucune hypothèse n'est formulée.

Les étapes suivantes visent à corriger certains détails de parcours avant de procéder à la validation des hypothèses :

```
RemoveTwinVolumes    (NewGeons) ;
ReplaceNullEdgeLoops (NewGeons) ;
```

La fonction `RemoveTwinVolumes` a pour but d'éliminer les hypothèses identiques, produites par la segmentation des passages. Tel qu'illustré à la figure 4.13, un passage est un trou ouvert à ses deux extrémités, qui débouche généralement sur deux surfaces opposées de l'objet, et dont les extrémités sont délimitées la plupart du temps par une boucle interne.

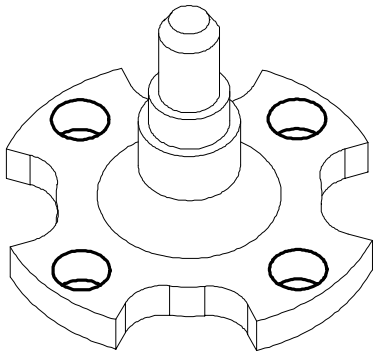


Figure 4.13 : Passages.

À ce titre, chaque boucle interne d'un passage déclenche une nouvelle hypothèse de géon, ce qui engendre finalement deux hypothèses pour le même géon négatif. Bien

qu'il aurait été possible de marquer progressivement les entités extraites pour éviter qu'elles ne soient choisies de nouveau par une autre hypothèse, il s'avère plus simple d'éliminer les doublets après coup.

Similairement, la fonction `ReplaceNullEdgeLoops` corrige un problème propre aux cônes et aux pyramides. Au cours d'une formulation d'hypothèse, la boucle interne est mémorisée à titre de profil générateur de départ, alors que le profil générateur d'arrivée est présumé correspondre aux arêtes qui provoquent l'arrêt du mécanisme d'extraction. Cependant, dans le cas des cônes et des pyramides, aucune arête terminale n'est disponible, l'extraction prenant fin par épuisement de surfaces ; ceci laisse donc un des profils générateurs indéterminé. Ce cas est facilement détecté par la fonction `ReplaceNullEdgeLoops`, qui cherchera alors à identifier le sommet spécifique qui correspond à l'apex du géon. En cas d'échec de cette procédure, l'hypothèse de géon sera éliminée, le programme considérant qu'il y a eu erreur de segmentation.

Vient ensuite l'étape de validation des hypothèses. Tel que nous l'avons évoqué au chapitre 3, le programme considère l'hypothèse valide s'il parvient à identifier le géon, c'est-à-dire s'il peut calculer les attributs du géon à partir des informations indiquées dans l'hypothèse :

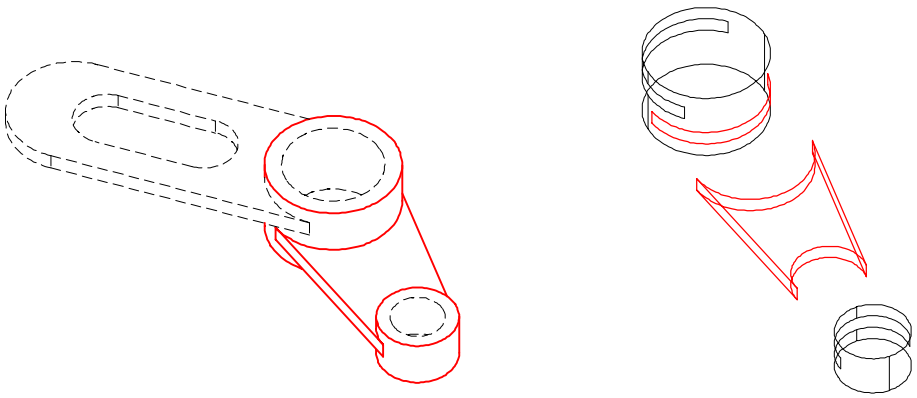
```
/*---------- Validation des hypothèses -----------*/
for (i = 0 ; i < NewGeons->NumEntries ; i++) {
   Status = ComputeGeonAttributes (TokenG) ;
   if ( Status == KBV_FAILURE )
      AltGeon = CheckForAlternateGeons (TokenG) ;
   ...
```

Nous n'entrerons pas ici dans le détail du calcul des attributs, qui sera expliqué plus loin ; il suffira de savoir pour l'instant que les attributs sont calculés géométriquement.

Tel que l'illustre l'extrait de code précédent, le programme amorce une tentative de recouvrement lorsque l'hypothèse n'est pas confirmée : il s'agit en l'occurrence d'essayer de préserver une hypothèse qui n'est peut-être pas totalement invalide. En effet, une de nos prémisses associe la boucle interne au profil générateur du géon ; ceci n'en demeure pas moins une présomption, dont la justesse ne se confirme qu'en règle générale.

Nous illustrons à la figure 4.14 une des pièces expérimentales qui fait justement exception à cette règle. On observe en 4.14 (b) qu'un des bras de l'objet peut effectivement être segmenté à partir de la boucle interne du cylindre central ; cependant, ce bras ne peut être généré que par la surface plane du géon.

Le cas échéant, la fonction `CheckForAlternateGeons` tentera donc d'interpréter différemment l'hypothèse initiale, en considérant toutes les possibilités de profil générateur dans le même groupe de surfaces. Chaque interprétation sera versée comme une nouvelle hypothèse, encore sujette à validation.



(a) (b)

Figure 4.14 : Boucle interne non-génératrice. (a) Vue solide de l'objet. (b) Vue en fil-de-fer d'une partie de l'objet. Ce type de représentation illustre les arêtes véritablement présentes dans la description CAO.

Si cette tentative de dérivation de l'hypothèse initiale porte fruit, l'interprétation correcte est substituée à l'hypothèse de départ ; dans le cas contraire, le programme rejette autant l'hypothèse initiale que ses avatars, et prend la décision de découper l'objet en deux parties distinctes là où la segmentation s'est amorcée, c'est-à-dire sur la boucle interne. Faute d'identification d'un quelconque géon, cette solution a l'avantage de faire évoluer le modèle vers quelque chose de moins complexe. La technique de découpage s'apparente aux mécanismes de complétion de modèle, puisqu'il suffit d'éliminer la boucle interne et d'ajuster une surface entre les arêtes d'origine.

Toutes les hypothèses valides seront finalement versées dans l'ensemble de mémorisation à titre de géons reconnus :

```
/*----- Déplacement des entités reconnues -----*/

CopyRecognizedEntities   (NewGeons) ;
RemoveRecognizedEntities (NewGeons) ;
```

Tel que mentionné auparavant, autant le géon que ses descendants sont transférés vers l'espace de mémorisation. Seules les entités qui appartiennent exclusivement à des géons reconnus sont transférées ; certaines arêtes limitrophes par exemple doivent être laissées dans l'espace d'analyse, puisqu'elles borderont des surfaces encore non-reconnues. Cette opération est donc effectuée en deux temps afin de transférer seulement les entités requises, et d'assurer une mise à jour correcte des pointeurs entre les entités non-reconnues.

Reste alors une dernière étape importante, la complétion de modèle, qui permettra de retrouver la validité géométrique de l'objet :

```
/*---- Complétion de modèle ----*/

CloseOpenLoops   () ;
CloseOpenVolumes () ;
```

Cette étape implique deux opérations principales, soit la fermeture des boucles, et l'ajout de surfaces. Considons par exemple le petit cylindre de la figure 4.14 (b), que nous illustrons de nouveau à la figure 4.15 (a).

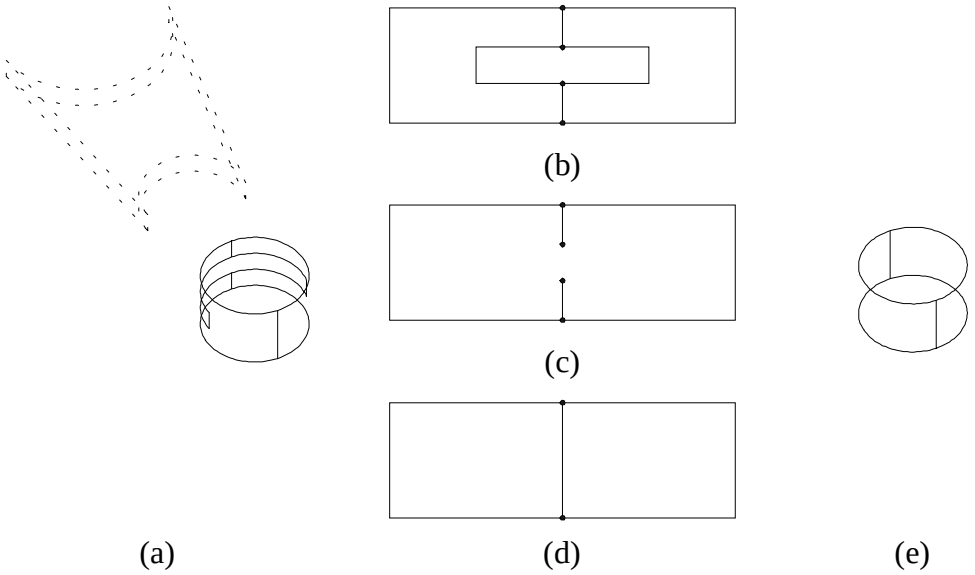


Figure 4.15 : Fermeture de boucles. (a) Vue en fil-de-fer de l'objet. (b) Boucles initiales des surfaces du cylindre, déroulées à plat. (c) Boucles modifiées par hypothèse. (d) Boucles finales, après complétion de modèle. (e) Cylindre résultant.

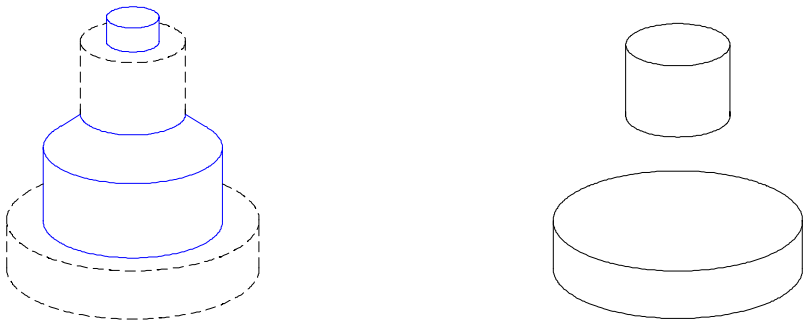
Lors du calcul des attributs, le programme aura rajouté une surface cylindrique aux deux extrémités du bras de la figure 4.15 (a). Cette opération visait à compléter le géon hypothétique, de sorte à permettre son identification formelle. La cohérence géométrique implique cependant de réassigner les arêtes des surfaces adjacentes, illustrées en 4.15 (b), aux nouvelles surfaces crées. Tel qu'on l'observe en 4.15 (c), les deux surfaces du cylindre deviennent alors invalides, puisque définies par des contours ouverts.

Bien qu'il serait possible de fermer les boucles immédiatement après la création des surfaces, ceci serait éventuellement dangereux. On doit comprendre en effet que la

manipulation se déroule *avant* la confirmation de l'hypothèse de géon, comme tentative justement de restaurer les entités manquantes qui font défaut pour l'identification. Dans l'éventualité où l'hypothèse serait fausse, le programme devrait alors rebrousser chemin, c'est-à-dire ramener la description sous sa forme originale. On comprend mieux alors la pertinence d'une modification partielle de la description, une modification complète s'avérant plus compliquée à défaire par la suite.

La fonction `CloseOpenLoops` détectera donc les boucles ouvertes et ajoutera une arête aux endroits requis. Une deuxième passe d'analyse vérifiera s'il y a lieu de fusionner certaines arêtes : il est clair en effet qu'en ajoutant une petite arête au milieu de la figure 4.15 (c), on obtient trois arêtes colinéaires, qui seraient plus utilement décrites par une seule grande arête. Le résultat de la fusion d'arêtes est illustré en 4.15 (d) ; remarquer d'autre part qu'il sera beaucoup plus facile par la suite de reconnaître le cylindre résultant, ses entités ayant évolué dans le sens d'une simplification descriptive.

La fonction `CloseOpenVolumes` répond au même genre de problématique, dans la mesure où l'extraction des géons reconnus laisse éventuellement certaines ouvertures béantes ; en témoigne par exemple la pièce 10, partiellement illustrée à la figure 4.16.



(a) (b)

Figure 4.16 : Fermeture de volumes. (a) Vue solide de l'objet, avec les géons reconnus en trait continu. (b) Entités non-reconnues.

Pour le cas illustré, l'algorithme parvient à reconnaître 3 des 5 géons, indiqués en trait continu à la figure 4.16 (a). Après l'identification du géon conique et l'élimination de ses descendants de l'espace d'analyse, le petit cylindre devient effectivement ouvert par en-dessous. Ce résultat est tout à fait normal puisqu'il n'existait initialement aucune surface à cet endroit, situé au coeur du solide plutôt qu'à sa périphérie.

Ce type de situation est facilement détecté, puisque une arête doit nécessairement être le lieu d'intersection de deux surfaces, alors qu'une arête béante ne possède qu'une seule surface adjacente. L'algorithme de fermeture de volumes doit donc repérer les différentes arêtes du modèle qui satisfont cette condition, les apparier ensemble de sorte à distinguer chacune des surfaces manquantes, puis finalement créer une surface dans l'espace délimité par chaque boucle d'arêtes.

4.3.3.2 Calcul des attributs du géon

Pour identifier le géon, le programme doit déterminer cinq attributs, à savoir le type d'arêtes génératrices, la courbure de l'axe, la symétrie du profil générateur, le type de fonction de balayage et la matérialité du géon.

Les deux premiers attributs peuvent être évalués assez directement. En effet, le type d'arêtes génératrices (droites, courbes ou hybrides) est obtenu par simple examen des arêtes du profil générateur, déjà identifié dans l'hypothèse. Compte tenu de la contrainte d'homogénéité imposée au domaine d'entrée, le géon n'est valide par ailleurs que si ses deux profils générateurs sont de même type.

Pour évaluer la courbure de l'axe, le programme doit identifier au préalable les arêtes latérales du géon, qui sont les arêtes reliant ensemble les deux profils générateurs. Cette identification s'effectue par exploration locale du modèle, en ne considérant que les arêtes indiquées par l'hypothèse. Compte tenu du domaine d'entrée, la courbure de

l'axe doit être de même nature que la courbure des arêtes latérales, tel qu'en témoigne la figure 4.17 ; cette règle serait cependant contredite en présence de surfaces sphériques ou de surfaces sculptées. Les propriétés des cylindres généralisés imposent d'autre part que toutes les arêtes latérales soient de même courbure.

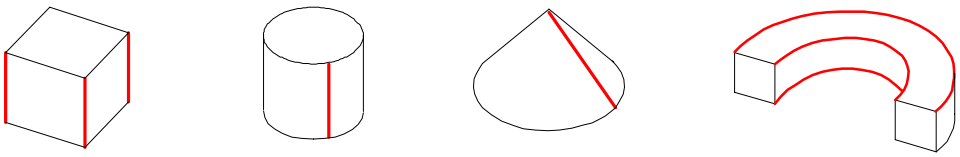


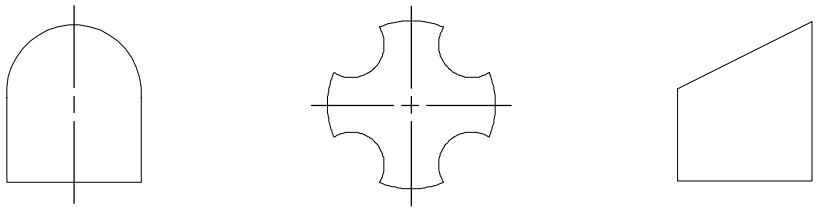
Figure 4.17 : Arêtes latérales.

Nous exploitons une autre propriété des arêtes latérales pour déterminer le type de fonction de balayage : la figure 4.17 indique effectivement que les géons à balayage constant présentent des arêtes latérales parallèles, ce qui n'est pas le cas pour les géons à balayage variable. Le recours à cette propriété présume évidemment que les arêtes latérales sont droites, puisque le parallélisme devient plus laborieux à vérifier sur des arêtes courbes. Nous n'avons pas tenu compte de ce cas particulier dans notre algorithme, étant donné l'absence de géons incurvés dans le domaine d'entrée. Une méthode plus générale cependant consisterait à comparer l'aire des profils générateurs polygonisés, identique pour un géon à balayage constant, et différente pour un géon à balayage variable. On doit noter cependant que ceci demeure vrai seulement en l'absence de surfaces sculptées.

Pour l'attribut de symétrie, rappelons que la symétrie est *réflective* si le profil générateur possède un seul axe de symétrie, *réflective et rotationnelle* s'il possède deux axes de symétrie perpendiculaires, et *aymétrique* s'il n'y a aucun axe de symétrie[15]. La figure 4.18 illustre ces différents cas.

---

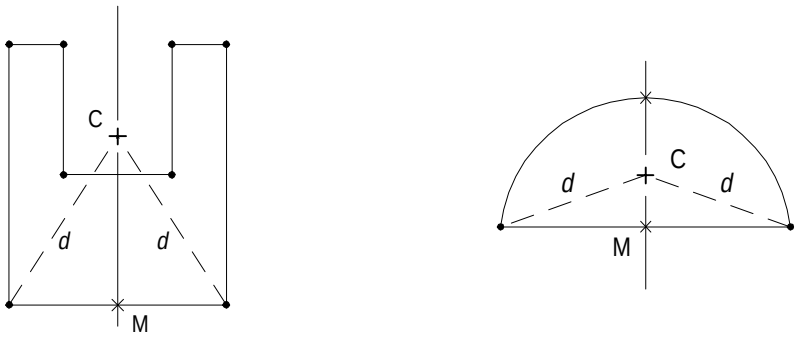
[15] Nous n'avons trouvé nulle part dans les textes de Biederman une définition claire et nette de la symétrie rotationnelle. Au sens commun, n'importe quel patron circulairement symétrique devrait être qualifié de rotationnel, tel un patron disposé à tous les 120°. La définition plus limitative que nous énonçons nous appartient, et répond seulement à des critères pratiques.



(a) (b) (c)

Figure 4.18 : Attribut de symétrie. (a) Symétrie réflective. (b) Symétrie réflective et rotationnelle. (c) Asymétrie.

La symétrie réflective est mesurée directement dans le plan porteur, par réflexivité d'une moitié du profil générateur sur l'autre moitié. Cette stratégie implique donc le calcul préalable de la position du ou des axes de symétrie potentiels, déterminés à partir des sommets des arêtes génératrices. Tel qu'illustré à la figure 4.19 (a), on observe que l'axe de symétrie passe nécessairement par le centroïde C du profil générateur, et par un deuxième point M situé à mi-chemin de sommets équidistants de ce centroïde.



(a) (b)

Figure 4.19 : Calcul de l'axe de symétrie d'un profil générateur.

Cet algorithme est donc susceptible d'identifier autant d'axes de symétrie potentiels qu'il y a de paires de sommets équidistants du centroïde. Les points milieu des arêtes sont aussi considérés dans le calcul, puisque les seuls sommets sont parfois

insuffisants pour déterminer l'emplacement de l'axe de symétrie. Pour le profil générateur de la figure 4.19 (b) par exemple, les points M et C seraient confondus en l'absence du point milieu de l'arc de cercle. L'utilisation conjointe des sommets et des points milieu des arêtes permet donc de polygoniser suffisamment le profil générateur, dans la mesure où la position estimée du centroïde tend vers sa position réelle. Une fois le ou les axes de symétrie localisés, reste à vérifier si tous les sommets situés d'un côté de l'axe se projettent perpendiculairement sur un sommet situé de l'autre côté de cet axe.

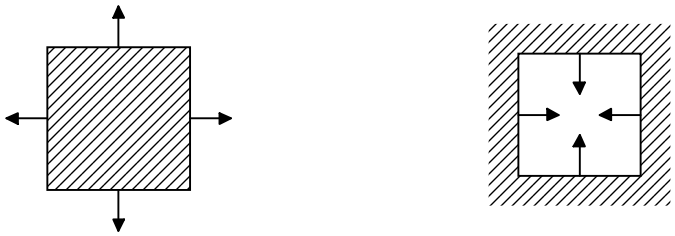
La symétrie rotationnelle est vérifiée similairement, en pivotant d'abord l'axe de symétrie réflexif de 45° dans le plan porteur, puis en tentant d'apparier les sommets de part et d'autre de ce nouvel axe. Cette vérification est cependant conditionnelle à la précédente, dans la mesure où l'absence de symétrie réflective exclut toute possibilité de symétrie rotationnelle[16].

Le profil générateur est déclaré asymétrique lorsque l'analyse des sommets n'indique aucun axe de symétrie potentiel, ou que la réflexivité des sommets sur tous les axes possibles s'avère infructueuse. Un géon n'est valide d'autre part que si ses deux profils générateurs présentent le même type de symétrie.

L'idée de base pour déterminer la matérialité du géon consiste à analyser les normales aux surfaces latérales. La figure 4.20 indique en effet que les normales extérieures à un géon solide pointent toutes vers des directions opposées, alors que les normales d'un géon négatif convergent vers le même espace. Il est par ailleurs impossible de fonder un jugement seulement sur l'analyse d'une ou deux surfaces latérales, la matérialité du géon résultant de l'interaction de la totalité des surfaces. D'autre part, faute de profils générateurs strictement convexes, il ne semble pas possible de recourir uniquement à la convexité des arêtes latérales comme critère de matérialité.

---

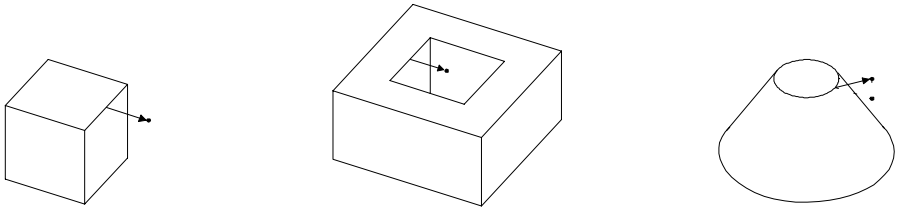
16 À cet égard, l'expression "symétrie réflective et rotationnelle" est pléonastique, puisque l'un est un cas particulier de l'autre.



(a) (b)

Figure 4.20 : Normales latérales de géons positifs ou négatifs. (a) Vue en coupe d'un géon positif. (b) Vue en coupe d'un géon négatif.

En pratique cependant, l'analyse des normales latérales se révèle laborieuse à mettre en oeuvre dans le domaine 3D, d'autant plus que la forme du profil générateur peut être très quelconque. Une technique beaucoup plus simple consiste alors à effectuer un test d'inclusion par rapport au profil générateur, tel qu'illustré à la figure 4.21.



(a) (b) (c)

Figure 4.21 : Matérialité déterminée par un test d'inclusion. Pour fins d'illustration, les points d'essai sont beaucoup plus loin que requis. (a) Géon positif. (b) Géon négatif. (c) Géon positif à balayage variable.

Il suffit en l'occurrence de calculer un point arbitraire à proximité d'une arête génératrice, dans la direction de la normale à une surface latérale, puis de vérifier si le point d'essai est situé à l'intérieur ou à l'extérieur du profil générateur. Cependant, ce test n'a de sens que si le point d'essai et le profil générateur appartiennent au même plan :

tel qu'illustré en 4.21 (c), le point d'essai doit être projeté dans le plan porteur du profil générateur pour un géon à balayage variable, faute de quoi le test serait inapplicable.

4.3.3.3 Calcul de convexité d'arête

Nous évoquons en terminant un dernier algorithme, celui qui sert à déterminer la convexité d'une arête. De nombreux auteurs recourent effectivement à la convexité d'arête dans l'analyse de données CAO, mais à peu près aucun ne daigne préciser la technique de calcul.

Rappelons qu'une arête est dite *convexe* si les surfaces adjacentes forment un angle solide de moins de 180°, et *concave* dans le cas contraire. Certains travaux admettent les catégories *doucement convexe* (*smooth convex*) et *doucement concave* (*smooth concave*) pour caractériser les arêtes où il n'y a pas de discontinuité abrupte de courbure, comme à la jonction de deux surfaces cylindriques. Nous n'avons pas eu à faire cette distinction dans notre projet.

La méthode générale pour déterminer la convexité d'une arête repose sur le produit mixte :

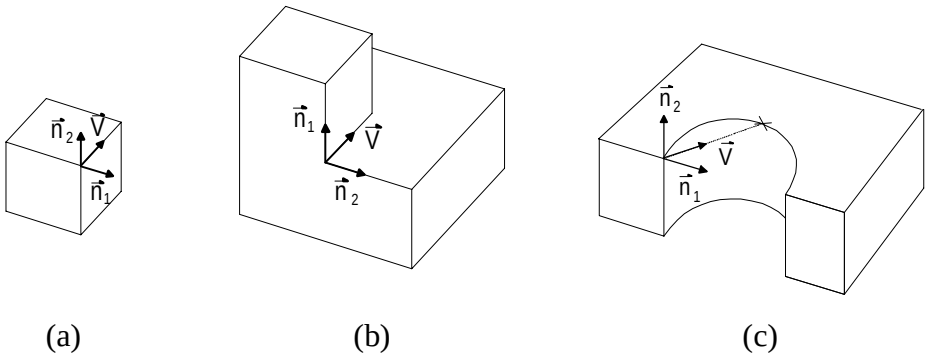
$$\text{si } (\vec{n}_1 \times \vec{n}_2) \cdot \vec{V} \begin{cases} < 0 & \Rightarrow \text{arête convexe} \\ > 0 & \Rightarrow \text{arête concave} \end{cases}$$

avec V : arête vectorisée dans le sens positif de parcours

n_1 : normale à la surface pour laquelle l'arête est de direction positive

n_2 : normale à la surface pour laquelle l'arête est de direction négative

Tel qu'illustré à la figure 4.22 (a) et (b), le produit vectoriel indique l'orientation relative des surfaces, alors que le produit scalaire permet de comparer ce résultat avec une référence locale stable, fournie par l'arête même.



(a) (b) (c)

Figure 4.22 : Calcul de convexité d'arête. Les figures présument un sens de parcours des arêtes selon la règle de la main gauche. (a) Arête convexe. (b) Arête concave. (c) Arête convexe courbe.

En pratique, ce principe général doit cependant être adapté aux différents cas particuliers. Si l'arête borde par exemple une surface cylindrique ou conique, le programme doit alors construire lui-même la ou les normales nécessaires au calcul. Tel qu'on l'observe en 4.22 (c) d'autre part, un arc de cercle doit être vectorisé depuis son point de départ jusqu'à son point milieu, afin d'éviter que le vecteur de référence V ne se confonde avec une des normales.

4.4 Conversion des données pour affichage (*TksToSat*)

4.4.1 Motivation

La pertinence de pouvoir visualiser les géons semble évidente dans le cadre de ce projet. Au cours du développement des programmes, nous devions analyser les résultats "manuellement", c'est-à-dire par examen des valeurs numériques dans un simple éditeur de texte. Cette démarche était difficilement évitable, puisque le déverminage d'un programme requiert une connaissance précise des données manipulées par l'ordinateur.

Cette façon d'examiner les résultats de *InnerLoopGeons* serait par contre très rebutante pour un utilisateur moyen, dans la mesure où elle est laborieuse, et implique

une connaissance approfondie des entités de l'objet et des règles de codification des données. À cet égard, la visualisation graphique apparaît comme une solution toute indiquée pour identifier rapidement les géons extraits de la description CAO.

Il est important de souligner ici que notre objectif n'est pas de mémoriser la description géon sous une forme définitive. En effet, dans la mesure où notre algorithme ne parvient pas à extraire tous les géons d'un modèle donné, il ne semble pas opportun à ce stade de générer immédiatement une description géon, qui serait de toute façon incomplète. À cet égard, le module *TksToSat* vise seulement à permettre la visualisation des données présentes dans les ensembles d'analyse ou de mémorisation.

4.4.2 Choix de l'environnement de visualisation

L'idéal aurait été de pouvoir visualiser les géons directement dans l'environnement d'exécution, c'est-à-dire dans l'interface de KBVision. Cet interface possède effectivement un module d'affichage permettant la visualisation d'entités tridimensionnelles. Deux facteurs nous ont cependant incité à rejeter cette solution :

- Le paradigme d'affichage 3D de KBVision repose sur la voxellisation, alors que nos entités sont décrites en Brep, c'est-à-dire algébriquement. En l'absence de fonctions dédiées dans la bibliothèque de KBV, il nous aurait donc fallu développer nous-mêmes les algorithmes de conversion des entités en voxels.

- L'autre difficulté concerne l'instabilité actuelle du module 3D. Nos essais indiquent en effet que cet outil plante souvent, en toutes sortes de circonstances. Qu'il s'agisse d'un problème d'installation spécifique au LIVIA ou d'un bogue non-corrigé du produit, le résultat reste le même, à savoir que le module 3D n'est pas vraiment fonctionnel.

Restait donc à considérer tout autre environnement graphique, opérant de préférence sur plate-forme Unix. De deux choses l'une :

1. Ou bien il nous fallait tout écrire de A à Z, depuis la conversion de données jusqu'à l'interface graphique en utilisant un *toolkit* quelconque, tel OpenGL, ACIS, DORE, etc., pour la manipulation d'entités, et Tcl/Tk, XMotif, etc., pour le développement de l'interface ;

2. Ou bien utiliser un logiciel de visualisation 3D déjà existant, et seulement traduire nos données sous un format de fichier CAO lisible par ce programme.

La première avenue offre la possibilité de concevoir un produit fait sur mesure, qui répond précisément aux attentes, et qui peut être raffiné à mesure que s'ajoutent de nouveaux besoins. À titre d'exemple, nous évoquerons le problème d'identification initiale des entités : l'interface de ProEngineer ne permet pas d'identifier l'étiquette de chacune des entités de l'objet à l'écran. Pour chaque nouvelle pièce, il nous fallait donc recouper manuellement l'information du fichier neutre, ceci à seule fin d'obtenir la "cartographie" de l'objet. Seul un produit maison pourrait offrir ce genre de fonctionnalité, pour le bénéfice du développeur. Principal inconvénient par contre, le temps de développement assez long pour parvenir à un produit acceptable, surtout au démarrage où tout est à faire.

Le peu de temps disponible étant notre principale contrainte, nous avons donc opté pour la deuxième solution. Nous avons d'ailleurs envisagé de reconvertir nos données en format neutre, quitte à n'utiliser de ProEngineer que la fonction d'affichage ; cette avenue était effectivement à considérer, surtout en raison de l'expérience acquise lors de la mise au point de *NeutralToTks*. Il appert cependant qu'il y aurait beaucoup de travail à faire pour ramener nos données sous leur forme initiale.

Bref, après étude des différentes possibilités, nous avons retenu le logiciel *ACIS 3D Viewer*, de Spatial Technology Inc., destiné strictement à la visualisation de données 3D. Plusieurs raisons motivent ce choix :

- Ce logiciel est relativement petit en terme d'espace disque (40 Megs), et gratuit. Diffusé publiquement depuis une dizaine d'années, le programme en est maintenant à sa 4e version, mais n'est disponible par contre qu'en version Windows.

- L'interface est doté des principales fonctions de manipulation graphique (rotation, translation, mise à l'échelle), et offre un rendu de qualité. Différentes fonctions permettent par exemple le contrôle des sources lumineuses, l'affichage solide, polygonisé ou fil-de-fer, etc. Le noyau manipule d'autre part des descriptions Brep, ce qui minimise notablement le travail de conversion de l'information.

- Contrairement à la plupart des modélisateurs, le visualisateur ACIS ne requiert aucune cohérence géométrique au sens d'Euler. Par exemple, il est tout à fait possible d'afficher une surface non-bornée, un volume ouvert, etc. Il s'agit d'un avantage manifeste pour ce projet, dans la mesure où l'extraction de géons pourrait conduire en principe à des résultats géométriquement incohérents. Le visualisateur ACIS permettrait malgré tout d'afficher ces résultats, contrairement à d'autres environnements plus stricts.

- Bien que le programme ne lise qu'un seul format de fichier CAO, le format .sat, ce format est solidement documenté [60], et assez simple à maîtriser minimalement. Il s'agit d'autre part d'un fichier ASCII, ce qui facilite la relecture manuelle pour le déverminage des programmes. Compte tenu de l'intérêt manifesté par l'industrie pour l'environnement ACIS, il est d'ailleurs probable que ce format devienne un standard *de facto* par la suite, au même titre que le format .dxf d'AutoCAD, largement utilisé maintenant pour l'échange de données 2D.

On notera que l'environnement ACIS proposé par Spatial Technology est d'abord et avant tout un noyau de modélisation en C++, utilisé au demeurant par plusieurs logiciels ou compagnies de renom. De plus, cet environnement est souvent évoqué dans les travaux antérieurs pour le développement d'interfaces adaptés au domaine de recherche.

Dans la mesure où il y aurait un intérêt soutenu au LIVIA pour l'interprétation de données CAO, l'environnement ACIS serait manifestement une acquisition à envisager. Le cas échéant, cette bibliothèque permettrait de développer un visualisateur adapté aux besoins, et portable sous Unix. Les données produites par la tâche *TksToSat* exposée ici seraient entièrement lisibles par ce produit maison, puisqu'elles respectent le format .sat intégré à l'environnement ACIS.

4.4.3 Paramètres d'entrée/sortie

La tâche *TksToSat* lit une description Brep sous forme d'éléments symboliques et convertit les données sous format .sat, version 4.0. L'unique fichier de sortie devra ensuite être soumis au logiciel *ACIS 3D Viewer* pour l'affichage graphique de l'information. Le travail requis pour la mise au point d'un convertisseur complet étant hors de notre portée, la tâche *TksToSat* ne convertit qu'une toute petite partie des entités ACIS admissibles, à savoir celles appartenant au domaine d'entrée de notre projet. La même observation s'applique d'ailleurs pour la tâche *NeutralToTks*.

Nous avons conçu le programme de telle sorte qu'il puisse servir à visualiser autant les géons que les entités non-reconnues. En effet, en l'absence de l'objet complet, l'affichage des seuls géons reconnus prêterait éventuellement à confusion, surtout si survient une erreur de segmentation de la part de *InnerLoopGeons*. Cette remarque est d'autant plus vraie que ce dernier programme n'est pas en mesure d'extraire tous les géons de l'objet, donc de restituer la forme globale à laquelle correspondent les géons identifiés.

À ce titre, la tâche peut recevoir en entrée un ou deux ensembles de données CAO :

- le premier ensemble est constitué de cinq fichiers, qui explicitent respectivement les volumes, surfaces, boucles, arêtes et sommets de l'objet ;

- le deuxième ensemble est formé de quatre fichiers, où apparaissent respectivement les surfaces, boucles, arêtes et sommets de l'objet. Les entités décrites dans cet ensemble sont présumées différentes de celles qui apparaissent dans le groupe précédent.

On comprend aisément que le premier groupe de fichiers est destiné à recevoir les ensembles de mémorisation produits par *InnerLoopGeons*, pour lesquels les géons constituent le plus haut niveau hiérarchique. Le deuxième groupe permettra d'autre part la conversion des entités restantes dans les ensembles d'analyse.

La tâche accepte autant de recevoir un seul ensemble que les deux à la fois : il serait donc possible de visualiser par la suite seulement les géons, seulement les entités non-reconnues, ou les deux à la fois. Le mécanisme de conversion n'utilisant par ailleurs que les seuls attributs de description (par opposition aux attributs d'interprétation) des éléments symboliques, il est tout à fait possible de convertir la description CAO produite par *NeutralToTks*, avant extraction des géons, pour visualiser l'objet complet.

Lors de la conversion, une couleur d'affichage unique est attribuée à chaque géon, alors que les entités non-reconnues sont colorées uniformément en gris. Ceci permet donc de distinguer rapidement les uns des autres lors de la visualisation.

4.4.4 Représentation ACIS des données CAO

Nous n'entrerons pas dans le détail du programme *TksToSat* puisqu'il s'agit fondamentalement d'une "moulinette", qui transcrit presque intégralement les données sources sous la forme prescrite par la norme SAT. À l'inverse de *NeutralToTks*, il n'y a aucune analyse de données à effectuer, sinon que certains changements de représentation mathématique.

La représentation ACIS est similaire à n'importe quel schéma Brep, c'est-à-dire que l'objet est décrit par le biais d'une structure hiérarchique qui précise la géométrie et la topologie de ses entités [61]. Nous illustrons à la figure 4.23 une instance simplifiée de hiérarchie ACIS.

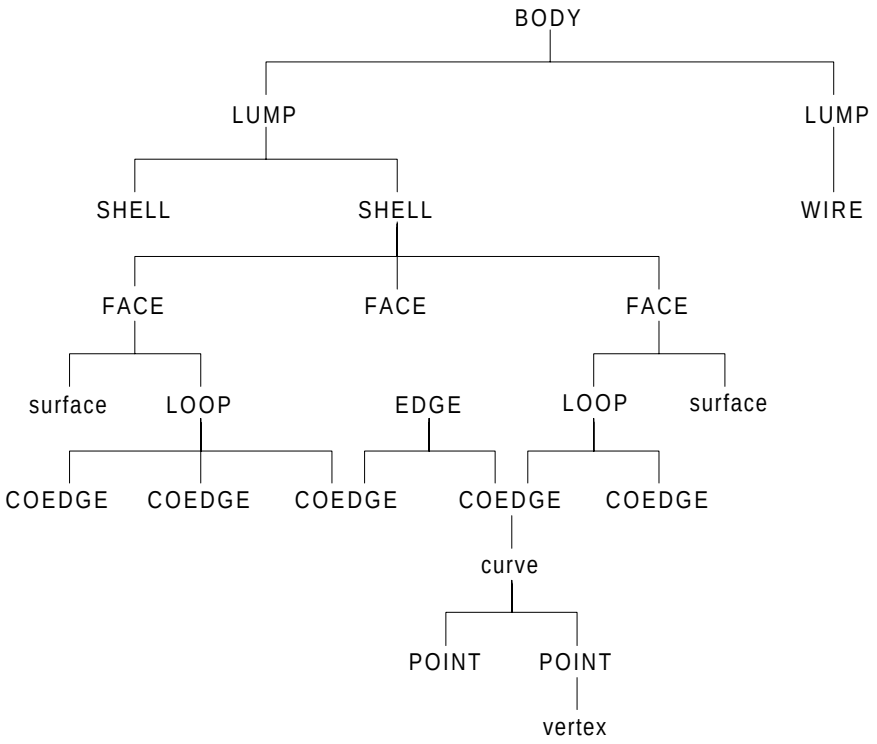


Figure 4.23 : Hiérarchie ACIS.

Tel qu'illustré, le type BODY représente le plus haut niveau d'abstraction du modèle en regroupant toutes les entités physiques de l'objet. Les LUMP permettent de distinguer les entités fil-de-fer des entités Brep, puisque l'un et l'autre peuvent coexister au sein de la même description.

Les types suivants s'apparentent beaucoup à ce que nous avons manipulé jusqu'à date : par exemple, le SHELL est un amalgame de surfaces voisines, représentées par des entités FACE ; lors de la conversion, chaque géon est d'ailleurs représenté par un objet SHELL. Nous n'irons pas plus loin dans cette énumération, notre intention étant seulement de mettre en évidence la similarité de nos données avec le schéma de représentation ACIS.

Nous mentionnons cependant deux différences avec notre modèle de représentation :

- Le sens de parcours des arêtes dans ACIS est à l'inverse de celui de ProEngineer, que nous préservons dans nos données. Ce passage d'une règle de main gauche à une règle de main droite implique donc que *TksToSat* doit intervertir la plupart des pointeurs entre les entités, faute de quoi l'objet apparaîtrait à l'écran retourné sur lui-même ;

- La représentation ACIS distingue explicitement chaque demie-arête, à l'aide des COEDGE, alors que nous mémorisons les deux sous un seul élément symbolique. Cette distinction ne pose en pratique aucun problème particulier de conversion, puisqu'il suffit de séparer l'information déjà existante.

Remarquons finalement que le schéma ACIS mémorise séparément une entité logique de sa représentation mathématique (FACE-surface, COEDGE-curve, POINT-vertex), bien que la seconde constitue plutôt un attribut de la première.

CHAPITRE 5

RÉSULTATS ET DISCUSSION

5.1 Résultats obtenus

La figure 5.1 résume le nombre de géons reconnus et non-reconnus pour l'ensemble des pièces expérimentales. Au total, la tâche *InnerLoopGeons* reconnaît correctement 52 % des géons, alors que 48 % d'entre eux restent encore à identifier.

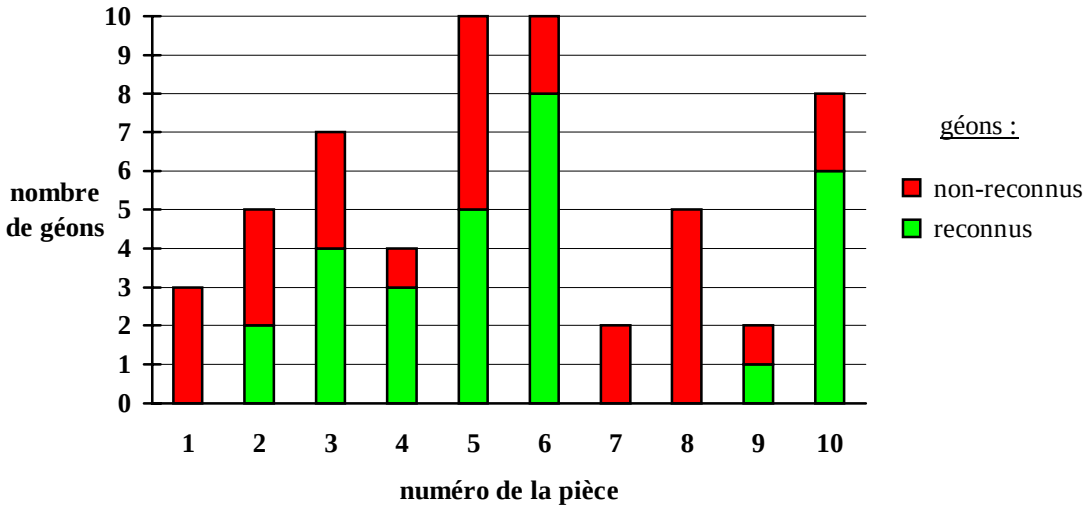
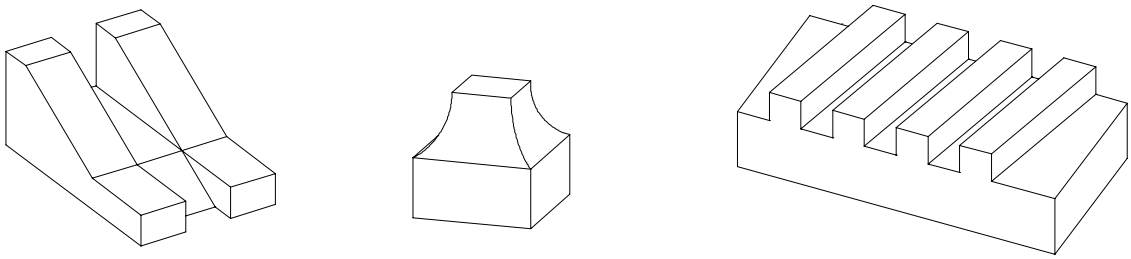


Figure 5.1 : Géons reconnus et non-reconnus pour l'ensemble des pièces.

Ces résultats doivent cependant être considérés avec un certain recul. En effet, la tâche *InnerLoopGeons* ne parviendra à identifier de géons qu'en présence de boucles internes sur le modèle ; ainsi, pour un autre ensemble de pièces, le même algorithme

aurait pu reconnaître plus ou moins de géons, selon le cas. Le nombre de géons reconnus explique cependant le choix initial de l'algorithme : dans la mesure où plusieurs techniques devront être appliquées pour géoniser entièrement les pièces expérimentales, nous avons évidemment retenu l'algorithme qui semblait le plus prometteur.

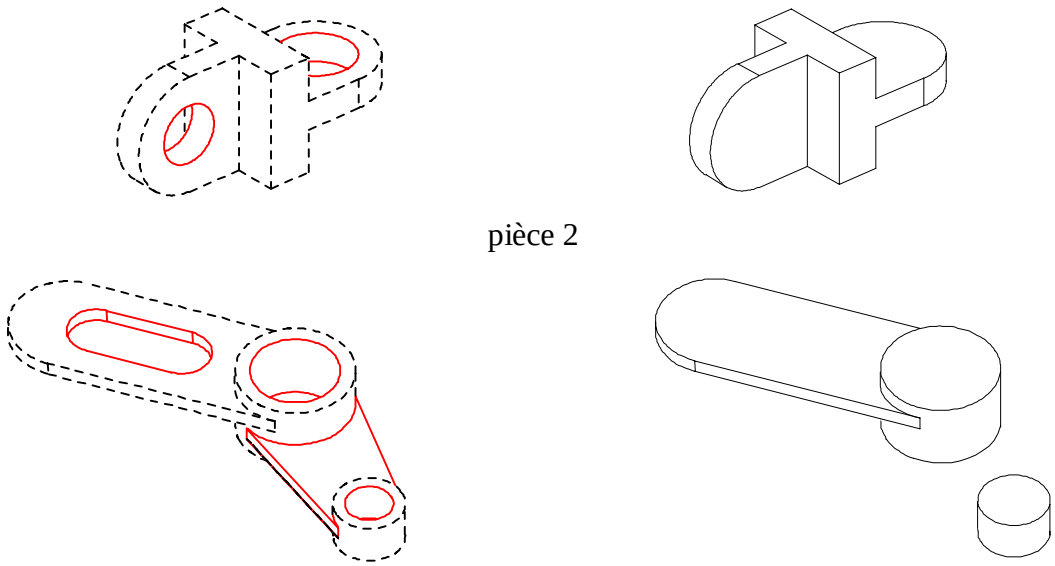
Il sera assurément pertinent d'examiner plus en détail les résultats pour chacune des pièces expérimentales. Tel qu'illustré à la figure 5.2, *InnerLoopGeons* est incapable d'identifier quelque géon que ce soit pour trois des pièces expérimentales ; l'absence de boucles internes sur ces objets explique ce résultat.



piece 1 **pièce 7** **pièce 8**

Figure 5.2 : Pièces n'ayant aucun géon reconnu.

La figure 5.3 illustre les résultats obtenus pour toutes les autres pièces. Les géons reconnus sont mis en évidence à gauche de la figure, alors que les géons qui restent à reconnaître sont illustrés à droite. Au total, 6 types de géons sont reconnus, ce qui représente 8.3 % du répertoire de 72 géons. Bien qu'il s'agisse apparemment d'une piètre performance, nous devons rappeler que les pièces expérimentales ne sont composées elles-mêmes que de 11 types de géons, tel que mentionné au tableau 3.4 (p. 80).



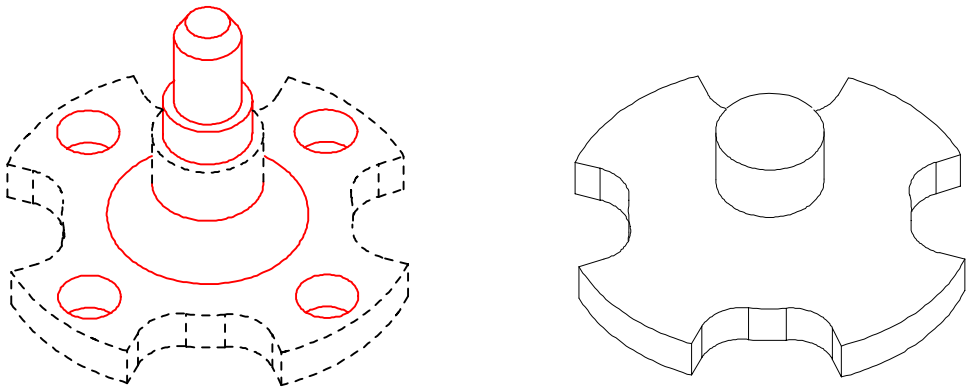
pièce 2

pièce 3

pièce 4

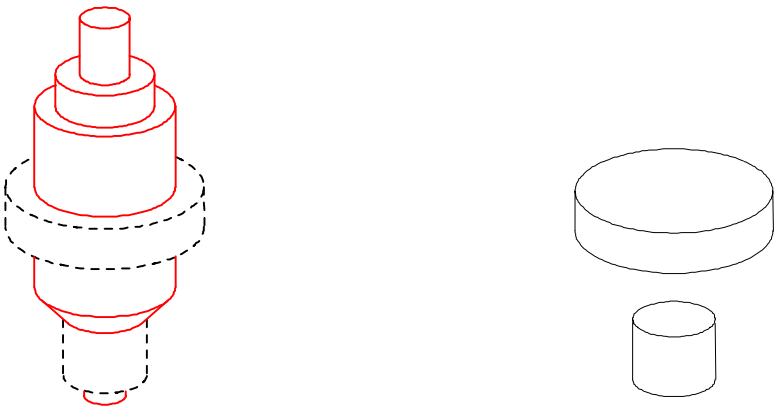
pièce 5

Figure 5.3 : Pièces dont les géons sont partiellement reconnus (début)



pièce 6

pièce 9



pièce 10

Figure 5.3 : Pièces dont les géons sont partiellement reconnus (fin)

5.2 Discussion

On constate à l'examen de la figure précédente qu'il y a une simplification manifeste des descriptions CAO après extraction des géons. Outre qu'il reste nécessairement moins d'entités à la fin du processus qu'au début, on observe d'autre part que le modèle s'épure progressivement de ses artefacts, pour laisser place aux entités plus globales ; cet effet est particulièrement flagrant pour les pièces 4, 5 et 6. Dans le sens de cette simplification, il faut aussi rapeller l'évolution de modèle illustrée au chapitre précédent pour la pièce 3, phénomène qui n'apparaît pas sur cette figure.

On pourra s'étonner au demeurant que le programme ait reconnu un seul des deux bras de la pièce 3 : ceci provient du découpage du cylindre central en deux demi-cylindres. En vertu du découpage effectué par ProEngineer, un des bras se greffe au cylindre central par le biais d'une boucle interne, alors que l'autre bras chevauche les boucles externes des deux demi-cylindres, tel qu'en témoigne la vue en fil-de-fer de cet objet à la figure 4.14 (p. 123). Seul le premier de ces bras est donc reconnu par notre algorithme, malgré la similitude morphologique entre les deux.

On observe aussi que l'algorithme laisse parfois comme résidu un ou deux géons facilement reconnaissables ; en témoignent les pièces 4, 6, 9 et 10. Notre lecteur s'étonnera peut-être que nous n'ayons pas tenté de les identifier immédiatement pour compléter la description géon de l'objet. Outre que le temps nous manquait pour ce travail, nous considérons aussi que cet objectif était moins essentiel, dans la mesure où il aurait été insuffisant pour finaliser le processus d'extraction de géons pour toutes les pièces. Cette problématique nous a plutôt incité à concentrer nos efforts sur l'algorithme central de notre projet.

La présence de géons séparés les uns des autres ne devrait poser aucun problème particulier pour le programme ultérieur. En fait, cette séparation de modèle simplifie le travail, puisque chaque groupe de surfaces délimite un espace d'analyse bien précis. Il

suffira au prochain programme d'identifier au préalable chaque sous-ensemble de surfaces, puis de concentrer son étude sur chacun des sous-groupes, comme s'il s'agissait d'objets distincts.

De façon plus générale, on note que la méthode utilisée est destinée à reconnaître le géon porté plutôt que le géon porteur, phénomène évident avec les pièces 4 et 9. En effet, considérant qu'une boucle interne soit vraisemblablement le point de départ d'un géon, la technique suppose alors une exploration du modèle qui exclue la surface dont provient la boucle interne, et par conséquent le géon auquel elle appartient. Ceci explique d'ailleurs pourquoi l'algorithme principal ne peut jamais reconnaître tous les géons d'un objet, puisqu'il excluera toujours de son analyse au moins le géon de base.

On doit aussi mentionner que l'application "pure et dure" de l'algorithme aurait consisté uniquement à séparer le modèle sur les boucles internes. En effet, la seule certitude qu'on puisse exprimer, c'est qu'une boucle interne représente le point de jonction d'un ou plusieurs géons avec le géon porteur. À cet égard, une approche conservatrice consisterait à reporter ultérieurement la phase d'extraction, pour simplement découper l'objet en éléments plus petits en présence de boucles internes. Cette approche serait justifiée à la longue par l'obtention de volumes toujours plus petits, qui tendraient à la limite vers le géon.

En principe, la séparation immédiate aurait l'avantage de délimiter par la suite des géons complètement fermés : notre méthode laisse effectivement plusieurs géons ouverts, qu'ils soient positifs ou négatifs. À titre d'exemple, considérons le cône de la pièce 9 : jamais notre programme n'ajoute de surface plane sous le cône pour fermer le volume. Il s'agit cependant d'un inconvénient très mineur, dans la mesure où l'application projetée requiert seulement d'obtenir la description géon de l'objet ; que les géons soient ouverts ou fermés demeure très accessoire, l'important étant de reconnaître le géon et d'identifier les entités originales qui le composent.

Tel qu'exposé auparavant, nous avons privilégié une approche plus agressive en générant au moins une hypothèse par boucle interne, la procédure de séparation n'étant utilisée qu'en dernier recours. Cette méthode permet d'autre part d'utiliser une information pertinente, à savoir la présence de la boucle interne qui sert à présumer d'un profil générateur, information qui serait autrement perdue après la complétion du modèle consécutive à la séparation immédiate.

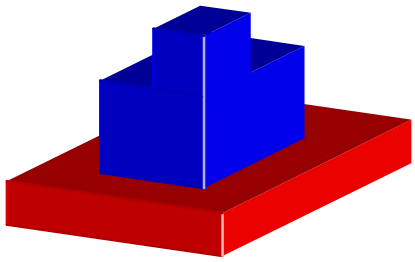


Figure 5.4 : Fouille en largeur d'abord.

On doit souligner par ailleurs que notre approche s'apparente à la fouille en largeur d'abord, puisqu'elle évite de trop s'aventurer dans une hypothèse. Par exemple, le programme serait actuellement incapable d'identifier les deux cubes supérieurs de la figure 5.4, bien que l'un d'entre eux soit ancré sur la boucle interne. Plutôt que de rejeter l'hypothèse et de séparer le modèle sur la boucle interne comme nous le faisons, il serait effectivement possible d'approfondir l'hypothèse, dans l'espoir d'arriver à distinguer les deux cubes par découpage des surfaces en L. Cette fouille en profondeur serait cependant prématurée dans la séquence d'analyse, puisque des opérations similaires de découpage doivent aussi être effectuées en l'absence de boucles internes.

À l'appui de notre propos, nous invitons le lecteur à comparer l'objet précédent qui contient une boucle interne, avec la pièce 5 illustrée à droite de la figure 5.3, qui ne contient plus de boucles internes. On constate que la situation géométrique est identique, à savoir qu'il y a un géon en bordure d'un autre, ce qui implique dans les deux cas un découpage de surfaces latérales.

Nous n'avons donc pas cherché à morceler plus avant une hypothèse invalide, puisque ce travail de découpage doit aussi être fait pour d'autres pièces, en dehors du contexte des boucles internes et avec d'autres techniques de segmentation. Il nous semblait plus efficace à cet égard d'amener nos données dans cet autre contexte, plutôt que d'importer ce contexte d'analyse dans le cadre de la validation d'hypothèse en cours.

5.2.1 Lacunes de l'algorithme actuel

Nous mentionnons en terminant certaines faiblesses de notre implémentation. Une première lacune concerne le mécanisme de manipulation d'hypothèse, trop peu évolué pour analyser des données incomplètes. Par exemple, lorsque une première hypothèse de géon s'avère invalide, notre programme dispose pour seule avenue de rechange de remettre en question la nature du profil générateur. Bien qu'il s'agisse effectivement d'une piste à explorer, celle-ci suppose par contre que la phase d'extraction ait identifié toutes les entités qui composent le géon, et seulement ces entités.

Bref, l'implémentation actuelle est totalement incapable d'élargir ou de restreindre une hypothèse ; tout au plus peut-elle modifier sa prémisse de profil générateur, mais sans jamais questionner les surfaces en cause. D'évidence, un mécanisme intelligent devrait affiner progressivement l'hypothèse, rejetant au besoin certaines entités pour essayer d'en extraire de nouvelles, plus conformes aux indices accumulés jusqu'à date.

L'outillage utilisé, le langage C, rend d'ailleurs plus laborieux l'implémentation de ce genre de mécanisme. À l'opposé des langages procéduraux, les langages symboliques permettent en effet de déclencher plus simplement d'autres fonctions à partir de nouveaux indices, d'explorer au passage des avenues imprévues, de revenir en arrière, etc. Il s'agit là en fait de la problématique typique des systèmes experts, où le programmeur n'est pas tenu de codifier le chemin qui mène à la solution, mais seulement les moyens possibles pour y parvenir.

Autre problème, celui de l'heuristique d'extraction des surfaces. À défaut d'une procédure capable de faire évoluer elle-même sa connaissance locale de l'objet, il faudrait donc pouvoir extraire immédiatement les bonnes surfaces, ce qui implique une heuristique fiable. Bien que notre heuristique s'avère entièrement fonctionnelle pour les pièces expérimentales, nous convenons qu'elle échoue pour d'autres objets qui n'ont rien d'exceptionnel en terme morphologique. La difficulté des heuristiques, c'est qu'il existe toujours des cas particuliers qui échappent au cas généraux qu'elles visent.

Reste aussi le problème des boucles hybrides, inaccessibles pour le programme actuel. Tel que mentionné au chapitre précédent, les boucles hybrides sont formées d'un enchaînement d'arêtes concaves et convexes ; la figure 5.5 illustre un objet de ce type.

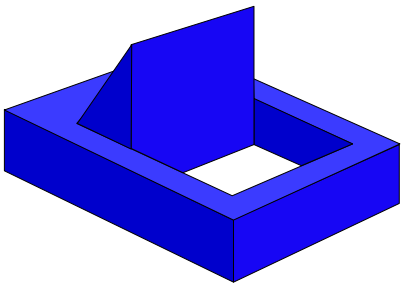


Figure 5.5 : Boucles hybrides.

Les boucles hybrides se distinguent des autres boucles internes par le nombre et surtout par le type de géons qu'elles délimitent. En effet, une boucle hybride circonscrit nécessairement plusieurs géons, dont certains seront positifs et d'autres négatifs ; à l'inverse, une boucle interne normale sera le lieu de jonction de un ou plusieurs géons, partageant cependant le même type de matérialité.

Si les boucles internes peuvent en principe être analysées globalement, les boucles hybrides doivent faire l'objet par contre d'une approche locale. En effet, on constate à l'examen de la figure précédente qu'une boucle hybride doit être analysée par morceaux, chacun d'entre eux correspondant à une séquence d'arêtes de même convexité.

Après fermeture des contours et extraction des surfaces adjacentes, il devient alors possible de générer les hypothèses de géon appropriées.

L'approche globale que nous appliquons provoquerait un échec complet d'analyse de ce genre de situation. Après l'extraction des surfaces latérales et l'invalidation de l'hypothèse de géon, le programme tenterait effectivement de séparer le modèle en ajustant une surface sur *toute* la boucle, sans tenir compte des entités situées à l'intérieur de cette boucle. Pour l'objet illustré à la figure précédente, ceci signifie qu'une surface plane serait ajoutée sans produire d'intersection avec la surface verticale. Il est évident ici que s'il faut ajouter un plan, ce dernier ne doit cependant fermer que le trou, et non la boucle hybride complète.

Actuellement, le programme rejettera un objet comportant une boucle hybride, situation facilement détectée après calcul de convexité d'arête. L'absence de boucles hybrides parmi les pièces expérimentales explique pourquoi nous n'avons pas travaillé ce type de situation. Le lecteur pourra consulter au besoin Corney et Clark [62] qui tentent de résoudre ce problème par une méthode de graphe.

CHAPITRE 6

TRAVAUX FUTURS

Nous étudierons brièvement dans ce chapitre les étapes nécessaires pour compléter l'analyse des pièces expérimentales. En dépit des lacunes soulignées au chapitre précédent, nous excluons ici toute analyse fondée sur les boucles internes, puisque l'implémentation actuelle s'avère suffisante pour les pièces du projet. Notre intention n'est pas de formuler un algorithme complet, mais seulement de mettre en évidence quelques pistes interessantes à explorer.

Nous avons déjà évoqué que les opérations antérieures de complétion de modèle produisent éventuellement des ensembles de surfaces séparés les uns des autres ; en témoignent par exemple les pièces 3 et 10 de la figure 5.3 (p. 143). Il nous apparaît inévitable que le prochain programme devra commencer par identifier chacun de ces sous-ensembles, de sorte à délimiter plus clairement les espaces d'analyse ultérieurs. La problématique des graphes suggère manifestement une fouille en profondeur du graphe des surfaces, implémentée assez facilement par une fonction récursive.

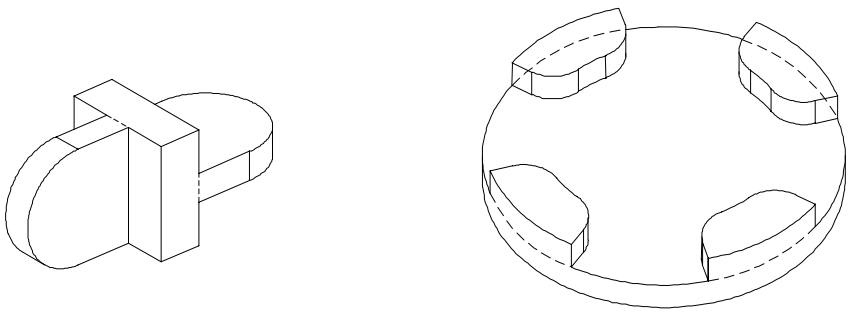
À ce stade, il devient très facile de terminer l'analyse des pièces 4, 6, 9 et 10, puisqu'elles ne comportent plus qu'un ou deux géons bien distincts, qui ne requièrent aucun découpage particulier. Au pis-aller, il serait effectivement possible d'appliquer un mécanisme de reconnaissance par défaut, qui consiste à générer une hypothèse de géon à partir de toutes les surfaces d'un sous-graphe. Le qualificatif "par défaut" réfère à la position logique de ce sous-programme : il devrait en principe apparaître à la toute fin, lorsqu'il n'y a plus aucun indice qui justifie une resegmentation des données.

L'ordinateur devrait alors supposer, par défaut, que chaque sous-graphe représente un seul géon. On peut souligner au passage que l'algorithme des boucles internes ne serait pas transposable au niveau des boucles externes pour identifier les géons restants. En effet, les boucles internes ne sont utilisables que parce qu'elles correspondent à une situation géométrique qui permet d'amorcer une recherche dans une direction précise. En contrepartie, les boucles externes ne fournissent aucun indice d'analyse particulier puisqu'elles circonscrivent chaque surface du modèle.

Toutes les autres pièces impliquent cependant un découpage du modèle en éléments plus petits, découpage qui doit être entrepris et guidé par des indices géométriques suffisamment prometteurs. Par exemple, le résidu des pièces 2, 3 et 5 suggère fortement d'amorcer certaines hypothèses de géon en présence de surfaces parallèles, présumées porteuses du profil générateur. Comme il peut exister au sein d'une description CAO des paires de plans parallèles qui ne correspondent à aucun géon, le programme ne devrait continuer son exploration que si les surfaces parallèles sont liées entre elles par un nombre suffisant d'arêtes, supposément latérales au géon.

On remarque par ailleurs que les arêtes concaves deviennent généralement le lieu où le géon hypothétique doit être segmenté du reste du modèle ; ce principe pourrait utilement être exploité pour l'extraction des surfaces et l'ajout des arêtes manquantes. On constate en effet qu'une part importante du travail d'analyse volumétrique découle de l'absence de certaines arêtes, nécessaires pour découper certaines surfaces qui chevauchent simultanément deux géons. À titre d'exemple, nous illustrons ces arêtes manquantes en pointillé à la figure 6.1.

La méthode générale que nous proposons pour la suite se résume donc dans ses grandes lignes aux éléments suivants :



(a) (b)

Figure 6.1 : Arêtes manquantes, illustrées en pointillé. (a) Résidu de la pièce 2. (b) Résidu de la pièce 5.

1. Déterminer toutes les paires de plans anti-parallèles du modèle. Deux plans sont anti-parallèles si leurs normales $\vec{n}_1$ et $\vec{n}_2$ respectent les conditions suivantes :

$$\vec{n}_1 \times \vec{n}_2 = 0$$
$$\vec{n}_1 \cdot \vec{n}_2 < 0$$

   Ce test simple permet d'effectuer une pré-sélection des surfaces planes susceptibles de constituer les deux extrémités d'un cylindre généralisé droit.

2. Pour chaque paire de plans anti-parallèles :

   2.1 Identifier les arêtes latérales qui joignent les plans entre eux.

   2.2 S'il n'existe aucune arête latérale entre les deux profils générateurs présumés, rejeter l'hypothèse ;

   2.3 S'il existe une arête latérale entre chaque sommet de l'une et l'autre plan, valider l'hypothèse. Notre lecteur pourra observer à la figure 5.3 (p. 143) que ceci se produit lorsque le sous-graphe détermine en fait un géon unique.

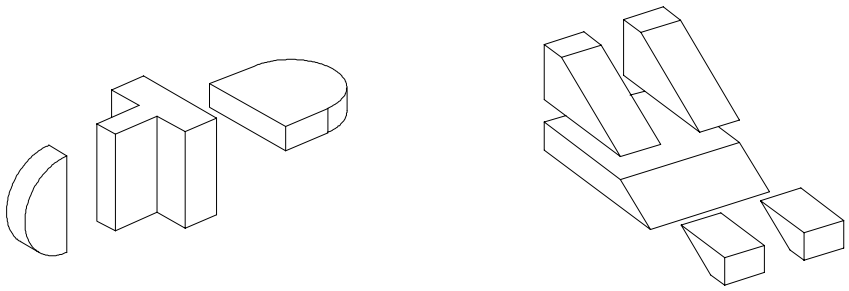
2.4 S'il existe moins d'arêtes latérales que de sommets sur l'un ou l'autre plan, découper une surface latérale en deux surfaces plus petites. La figure 6.1 illustre deux cas différents : en (a), il faut lier les deux profils générateurs en ajoutant des arêtes, ce qui suppose un morcellement des surfaces en T. En (b) par contre, il faut fermer un profil générateur, ce qui provoque aussi un découpage de la surface cylindrique latérale. Les deux situations se distinguent facilement ici, dans la mesure où les profils générateurs présumés comportent ou non le même nombre de sommets.

Les plans parallèles devraient aussi faire l'objet d'une analyse semblable. En effet, on observe à la figure 6.1 (b) que les surfaces supérieures des protrusions sont parallèles avec la surface supérieure du géon de base ; bien que cette dernière n'appartienne pas aux protrusions, elle délimite très clairement par contre l'extension des protrusions. Il y aurait une simplification manifeste de l'analyse, puisqu'il faut ajouter dans ce cas une seule arête manquante pour chaque hypothèse de géon, alors que la méthode anti-parallèle, qui s'interesserait au disque, requiert l'ajout de quatre arêtes pour la même hypothèse. Le reste de la procédure reprend essentiellement le cheminement usuel, soit la validation des hypothèses, la complétion du modèle, et la réitération du processus pour les entités restantes.

Remarquons au passage que la complétion de modèle introduira des opérations de fusion de surfaces, dont nous n'avons pas eu usage dans nos travaux. Par exemple, après extraction de chaque petit géon de la pièce 8, il faudra nécessairement fusionner les morceaux de la surface supérieure du grand géon. Cette opération ne devrait pas poser de grandes difficultés si les deux surfaces latérales du grand géon ont déjà été découpées aux points de jonction des petits géons, tel que le suggère notre algorithme.

On doit souligner par contre que la procédure des plans anti-parallèles n'est pas sans entraîner certains effets indésirables. Cet algorithme pourrait par exemple découper

la pièce 2 tel qu'illustré à la figure 6.2 (a), éventualité qui dépend de l'ordre d'apparition des données dans la description. Ce découpage, qui n'engendre pas du tout les géons désirés, devrait idéalement être filtré avant qu'il ne se produise, à l'aide d'une condition qui reste à déterminer.



(a) (b)

Figure 6.2 : Effets secondaires de la géonisation par plans anti-parallèles. (a) Résultat possible pour la pièce 2. (b) Résultat anticipé pour la pièce 1.

La même remarque s'applique aussi pour la pièce 1, dont le découpage incontrôlé menerait en principe aux géons illustrés en 6.2 (b). Pour ce cas, on observe par contre que le découpage affecte des surfaces présumées génératrices, ce qui serait une condition valable pour en interdire l'occurrence. On doit noter par ailleurs que la véritable difficulté pour géoniser la piece 1 réside dans l'identification des surfaces génératrices des géons latéraux. Outre que chacune des surfaces internes de ces géons est découpée en deux par le géon central, on constate aussi qu'il manque non pas une, mais deux arêtes pour reconstituer le profil générateur. Nous croyons que notre successeur devrait réserver cette pièce de choix pour la fin, lorsqu'il aura acquis suffisamment d'expérience avec la géonisation des autres pièces.

Mentionnons finalement que l'algorithme des plans anti-parallèles permettrait aussi de reconnaître le parallélépipède de la pièce 7. Tel que nous l'avons mentionné auparavant, la présence de splines sur cette pièce risque cependant de compliquer

l'identification de l'autre géon. Il appartiendra donc à notre successeur de choisir la marche à suivre parmi trois solutions possibles :

1. Une solution assez immédiate pour fin de validation consisterait à approximer chaque spline par une arête droite ou un arc de cercle[17]. La portée réelle d'une telle méthode reste évidemment sujette à caution, puisque la validité du principe ne pourra pas être démontrée avec d'autres pièces.

2. L'autre hypothèse serait d'accepter la présence de splines, ce qui impliquerait alors de modifier *NeutralToTks*, puis de développer l'outillage d'analyse pertinent à ce genre d'entité.

3. Le dernier choix serait de rejetter totalement les splines dans le domaine d'entrée, donc de refuser la pièce 7. Cette dernière hypothèse serait acceptable dans le mesure où notre domaine n'est pas soumis aux besoins d'une application industrielle précise, et que l'analyse de splines pose des difficultés importantes. Nous ne recensons d'ailleurs aucun travail antérieur qui admettait les splines dans la description CAO d'entrée.

Une solution plus drastique serait de remplacer la pièce 7 par une autre pièce morphologiquement semblable, par exemple celle illustrée à la figure 6.3, qui n'utilise que des plans. Nous croyons que cette solution serait de loin préférable au maintien de la pièce 7 originale. D'une part, les splines sont trop marginaux dans ce projet pour prétendre à une quelconque affirmation scientifique ; d'autre part, une motivation purement exploratoire du sujet risque de coûter un travail disproportionné par rapport au reste du projet. Un domaine plus restreint permet par contre une meilleure validation des algorithmes développés.

---

[17] L'examen du fichier neutre révèle en fait que la pièce 7 est composée de 26 surfaces, planes ou cylindriques, et de 71 arêtes, lignes, arcs et splines confondus. L'analyse géométrique s'annonce plus complexe qu'elle ne paraît au vu de l'objet.

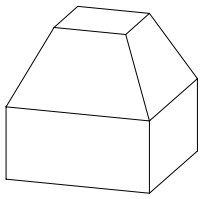


Figure 6.3 : Substitut pour la pièce 7.

Quant au calcul de connectivité qui doit permettre de finaliser la description géon des pièces, nous suggérons fortement que soit envisagée la deuxième proposition de codification mentionnée au § 3.1.4, nettement plus simple à implémenter en 2D et en 3D que la proposition préconisée actuellement. Une réflexion devra cependant être menée au préalable pour s'assurer que cette proposition offre un pouvoir d'indexation suffisant, et qu'elle ne provoque pas de descriptions conflictuelles parmi les pièces existantes.

Considérant finalement les différents besoins du module CAO→géons, nous prévoyons qu'il faudra regrouper toutes les tâches dans un ensemble unique, illustré à la figure 6.4. Tel que mentionné au chapitre précédent, seules les tâches *NeutralToTks*, *InnerLoopGeons* et *TksToSat* sont actuellement programmées, les autres tâches illustrées constituant seulement une proposition pour structurer le module.

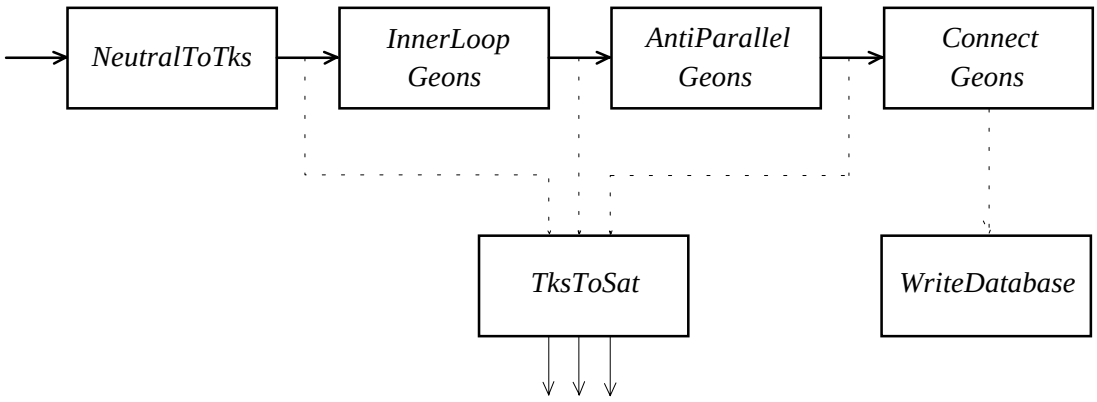


Figure 6.4 : Module *CadToGeons.*

Pour résumer ce module :

- *NeutralToTks* convertit un fichier neutre en une description CAO simplifiée.

- *InnerLoopGeons* extrait les géons à partir des boucles internes du modèle.

- *AntiParallelGeons* extrait les géons selon la méthode proposée dans ce chapitre, ou toute autre méthode jugée plus efficace.

- *ConnectGeons* procède au calcul de connectivité. Il est préférable de séparer cette fonction des programmes d'extraction de géons au cas où de nouvelles pièces s'ajouteraient plus tard au domaine d'entrée. Le cas échéant, il faudrait probablement poursuivre l'analyse en ajoutant d'autres programmes en aval de *AntiParallelGeons*, d'où l'intérêt d'une fonction distincte, facilement déplaçable.

- *TksToSat* convertit une description CAO symbolique sous un format permettant la visualisation des données. Une tâche qui pourrait afficher les données directement dans l'environnement KBVision, ou sinon sur plate-forme Unix, serait cependant préférable.

- *WriteDatabase* écrit une description géon dans la base de données du projet global de métrologie industrielle. Cette fonction ne doit pas se déclencher automatiquement après le calcul de connectivité, puisque la validation des résultats implique une nécessaire supervision humaine.

CONCLUSION

Il sera sans doute pertinent de rappeller ici les principales étapes de notre démarche, qui avait pour objectif premier la reconnaissance de géons dans une description CAO.

La première étape de notre réflexion a consisté à revoir les attributs du géon qui, dans leur forme originale, s'adaptent mal à la problématique de données CAO. Nous avons donc proposé trois modifications, à savoir :

1. L'introduction de géons négatifs, nécessaires pour décrire concisément les cavités ;
2. La simplification de l'attribut de dimension, qui élimine la distinction de croissance ou décroissance d'un géon en admettant plus simplement la dimension *variable* ;
3. Et l'ajout d'une nouvelle catégorie de profil générateur, constitué d'arêtes droites et courbes, dit profil *hybride*.

La théorie de Reconnaissance par composantes ne prévoyant aucun principe ou mécanisme particulier pour définir les géons dans un corps, nous avons procédé ensuite à la définition fonctionnelle des géons pour les pièces expérimentales. Face à la difficulté de formuler une loi de géonisation, cette approche consistait à définir *a priori* le résultat désiré de la géonisation pour chacune des pièces.

Le coeur du problème consistait ensuite à développer un algorithme d'extraction de géons. Devant l'impossibilité de mettre au point un algorithme unique, capable d'extraire tous les géons d'un objet, nous avons choisi celui qui permettrait à tout le moins de reconnaître un maximum de géons. Considérant les pièces expérimentales

choisies pour le projet, nous avons retenu la méthode de séparation sur boucle interne, qui permet d'identifier la moitié des géons de toutes les pièces.

Cet algorithme recherche dans la description CAO une certaine condition géométrique, qui sert d'indice de départ à l'analyse ultérieure. Un problème crucial de toute analyse CAO consiste effectivement à déterminer un point de départ valable dans les données afin d'éviter une recherche tous azimuts, aux conséquences souvent imprévues. Ayant alors localisé une boucle interne sur l'objet, cet algorithme procède ensuite par exploration locale des surfaces voisisnes, et valide les hypothèses résultantes par calcul géométrique, à l'aide des arêtes du géon hypothétique. Si l'hypothèse et ses variantes s'avèrent invalides, l'algorithme prévoit finalement un découpage de l'objet sur la boucle interne, ce qui permet au moins de faire évoluer la connaissance de l'objet, faute de géonisation.

* * *

On doit souligner que cette tentative d'importer le concept de géon dans l'analyse de données CAO demeure une première, à notre connaissance. En effet, malgré les affinités évidentes de notre projet avec l'extraction de caractéristiques, nous ne recensons aucune tentative de décrire un objet manufacturé quelconque à l'aide d'une description géon. Les essais antérieurs ont toujours porté sur la géonisation d'objets morphologiquement simples, dans un contexte de vision artificielle. Ce projet embrasse un domaine nettement plus vaste que celui considéré jusqu'à date par les approches géons.

Notre contribution au domaine demeure par contre assez marginale. En effet, les modifications proposées aux attributs du géon ont peu d'impact théorique, alors que le concept de géon négatif est simplement importé du génie mécanique. Bien que cette introduction soit nouvelle pour le domaine, elle ne révolutionnera en rien l'approche géon, sinon que de permettre un vocabulaire plus étendu.

L'essentiel de l'algorithme d'analyse de données demeure quant à lui assez conventionnel. Les boucles internes représentent en effet une situation géométrique commune en modélisation, et s'avèrent relativement simples à détecter. À cet égard, elles ont souvent été utilisées pour amorcer des algorithmes semblables au nôtre, et constituent fréquemment la première source d'information exploitée dans ce genre de travail.

Au niveau théorique, l'absence d'une définition formelle du géon nous apparaît comme une lacune importante de la théorie de Reconnaissance par composantes. Bien que ceci permet de définir un géon adapté au contexte, reste cependant la difficulté de parvenir à une définition cohérente, et dont l'application produise une solution unique et indépendante du point de départ.

En fait, la théorie de Reconnaissance par composantes nous semble périlleuse à utiliser pour l'identification d'objets manufacturés *quelconques*, tels que ceux visés par ce projet. La plus grande difficulté à laquelle nous avons été confronté concerne l'instabilité de ce qui est recherché : le géon est une entité perceptuelle, alors qu'un ordinateur requiert des critères d'analyse stricts et clairement définis. En pratique, il s'avère très difficile de traduire le mécanisme de découpage effectué par l'esprit humain en termes informatiques, d'autant plus que ce découpage est assorti d'une certaine part de subjectivité.

Nous avons déjà évoqué ce problème à la figure 3.9 (p. 78). S'il devrait définitivement y avoir une rainure dans l'objet de la fig. 3.9 (b), pourquoi n'y a t-il pas de rainures alors dans l'objet de la fig. 3.9 (a) ? Quelle que soit la réponse à cette question, on doit surtout observer ici qu'une même situation géométrique (un groupe de trois plans perpendiculaires entre eux) produit pourtant deux solutions différentes, selon l'objet auquel appartient la caractéristique. Cette multiplicité de *définitions* est très particulière

à l'approche géon, dans la mesure où les techniques d'analyse usuelles n'associent jamais qu'une seule solution à une même condition initiale des données.

D'évidence, la géonisation d'un corps repose autant sur une analyse locale que globale de l'objet, même si notre implémentation se limite strictement à l'analyse locale. Faute de pouvoir expliciter des critères infaillibles de géonisation, nous avons contourné ce problème de définition du géon en proposant une solution *a priori* pour chacune des pièces expérimentales ; cette méthode n'en demeure pas moins risquée pour le futur, puisque rien ne garantit une convergence des modules 2D→géons et CAO→géons pour de nouvelles pièces. En effet, compte tenu de cette stratégie, il sera possible à force de travail de faire en sorte que ces modules convergent tous deux vers la solution désirée pour chacune des pièces ; cependant, en l'absence d'un critère général de géonisation pour une pièce quelconque, toute nouvelle pièce risque d'être interprétée différemment par les programmes existants. À moins de pouvoir formuler une véritable règle de géonisation des objets, nous pressentons que ce dilemme ne pourra être utilement résolu que par le recours aux techniques d'intelligence artificielle, où l'ordinateur apprendrait par l'exemple.

Nous croyons d'autre part que le problème de multiplicité descriptive pourrait s'avérer la pierre d'achoppement du projet global de métrologie industrielle. En effet, l'avenue explorée actuellement suppose l'unicité descriptive des objets ; outre que cette prémisse se révèle plutôt contredite par nos modestes travaux, il semble aussi que la multiplicité descriptive soit intrinsèque à l'approche géon. Rappelons au passage que la multiplicité descriptive se concrétise sous deux formes différentes :

1. Le découpage d'un objet de plusieurs façons différentes, ce qui génère autant de descriptions géon ;
2. L'interprétation variable de certains cylindres généralisés, tel qu'illustré à la figure 1.5 (p. 14) ou à la figure 4.10 (p. 113).

Il nous semble que le projet global devra tôt ou tard admettre la multiplicité descriptive, en mémorisant plusieurs descriptions géon pour un même objet. Il y a d'ailleurs une similitude frappante avec le concept des vues caractéristiques, couramment utilisé en vision pour mémoriser les différentes instances visuelles d'un corps dans une image. S'il tombe sous le sens qu'il existe plusieurs possibilités d'observation d'un objet, il semblerait conséquent de reconnaître qu'il en existe aussi plusieurs interprétations. Cette avenue diminuerait de beaucoup la charge sur le module 2D→géons, qui n'aurait plus qu'à reconnaître n'importe quelle description géon d'un objet, mais augmenterait par contre le travail du module CAO→géons, qui devrait alors produire plusieurs descriptions géon possibles d'un objet.

Nous soulignerons d'autre part que si les géons négatifs s'avèrent utiles pour décrire certains types d'entités, les règles qui gouvernent leur utilisation efficace dans le contexte CAO demeurent encore très imprécises. Par exemple, nous refusons dans ce projet un géon négatif en bordure de l'objet ; en vertu de cette contrainte fonctionnelle, il deviendrait alors très laborieux de géoniser un simpe cube avec un coin coupé. La réflexion devra donc être poursuivie afin de mieux délimiter les conditions d'usage de ces géons.

Notre dernière remarque concerne l'appréciation personnelle que nous inspirent nos propres travaux. Nous avons été étonné de constater à quel point un problème d'apparence simple se révèle finalement autrement plus complexe et subtil qu'il ne semblait à prime abord. Après cette expérience, nous comprenons nettement mieux une certaine problématique de l'intelligence artificielle – dont ne se réclame d'ailleurs pas ce travail – et qui cherche à résoudre des problèmes tout à fait triviaux pour l'entendement humain. Nous avons appris aussi l'immense décalage qui existe entre un principe, toujours simple, et sa mise en oeuvre, beaucoup plus laborieuse et pleine d'imprévus.

En dépit d'un retard assez manifeste entre notre travail et l'état de l'art en matière d'analyse de données CAO, nous soutenons qu'il demeure défendable, au sens où un certain problème est résolu, peut-être imparfaitement, mais de façon ni pire ni meilleure que ce qui s'est fait à date pour des projets débutant en la matière. Nous invitons notre lecteur à revoir au besoin la figure 2.7 (p. 54), qui illustre les caractéristiques reconnues dans un projet qui date tout juste de 1997 ; nous n'avons somme toute rien à envier aux autres débutants.

D'un point de vue scientifique, nous soutenons aussi que ce travail n'a finalement qu'une valeur très accessoire. D'une certaine façon, ce travail retombe dans les mêmes ornières que tous nos prédecesseurs, celles du cas particulier et de l'heuristique. En effet, quelles que soient les techniques de validation mises en oeuvre, la segmentation des données n'en demeure pas moins le noeud gordien du problème. La segmentation implique invariablement une étude des cas particuliers dont on déduit un cas général, aux propriétés relativement stables et donc manipulable par heuristique, mais qui exclut finalement d'innombrables autres cas particuliers. Outre l'effort considérable qu'il faut déployer pour contrôler la déviation indésirable de l'heuristique, cette approche s'avère surtout interminable, étant donné l'infinité de cas particuliers qui restent hors de portée.

Nous pressentons depuis longtemps que le domaine d'entrée devrait d'abord être transformé avant analyse vers un domaine plus significatif que l'univers géométrique, lieu où se déroule finalement l'essentiel de toute analyse. Peut-être un champ vectoriel de normales ? Une transformation mathématique du genre Hough ? Beaucoup de pistes restent encore à explorer.

BIBLIOGRAPHIE

1. Binford, T.O. (1971). Visual perception by computer. *Proc. of IEEE Conf. on Systems Science and Cybernetics*, Miami, FL.
2. Rogers, D.F., Adams, J.A. (1990). *Mathematical elements for computer graphics* (2e éd.). McGraw-Hill.
3. Biederman, I. (1985). Human image understanding : recent research and a theory. *Computer Graphics, Vision, and Image Processing*, **32**, pp. 29-73.
4. Biederman, I. (1987). Recognition by components : a theory of human image understanding. *Psychological Review*, **94**, pp. 118-122.
5. Dickinson, S., Pentland, A., Rosenfeld, A. (1992). 3D shape recovery using distributed aspect matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **14**(2), pp. 174-198.
6. Jacot-Descombes, A., Pun, T. (1992). A probabilistic approach to 3D inference of geons from a 2D view. *Proc. of SPIE Applications of Artificial Intelligence X : Machine Vision and Robotics*, Orlando, FL, pp. 579-588.
7. Raja, N., Jain, A. (1992). Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, **10**(3), pp. 179-190.
8. Bergevin, R., Levine, M. (1992). Part decomposition of objects from single view line drawings. *CVGIP : Image understanding*, **55**(1), pp. 73-83.
9. Bergevin, R., Levine, M. (1993). Generic object recognition : building and matching coarse descriptions from line drawings. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **15**(1), pp. 19-36.
10. Du, L., Munck-Fairwood, R. (1995). Geon recognition through robust feature grouping. *Proc. of 9th Scandinavian Conf. on Image Analysis*, Uppsala, Suède.
11. Hummel, J., Biederman, I. (1992). Dynamic binding in a neural network for shape recognition. *Psychological Review*, **99**(3), pp. 480-517.
12. Pilu, M., Fisher, R.B. (1997). *Recognition of geons by parametric deformable contour models*. Rapport technique, Dép. d'Intelligence Artificielle, Université d'Edinbourg.
13. Wu, K., Levine, M.D., (1993). *3D object representation using parametric geons*. Rapport technique TR-CIM-93-13, Centre For Intelligent Machines, Université McGill, Montréal.
14. Madsen, C.B., Kirkeby, N.O.S., Christensen, H.I., (1991). A graph based approach to 3D qualitative scene modelling. *Proc. of 7th Scandinavian Conf. on Image Analysis*, Aalborg, Danemark, pp. 324-337.

15. Nguyen, Q.L., Levine, M.D., (1996). Representing 3-D objects in range images using geons. *Computer Vision and Image Understanding*, **63**(1), pp. 158-168.
16. Chin, R., Dyer, C. (1986). Model-based recognition in robot vision. *ACM Computing Surveys*, **18**(1), pp. 67-108.
17. Shah, J., Rogers, M. (1988). Expert form feature modeling shell. *Computer-Aided Design*, **20**(9), pp. 515-524.
18. Shah, J., Mäntylä, M. (1995). *Parametric and feature-based CAD/CAM*. Wiley & Sons.
19. International Standard Organization (1990). *STEP Form features information model, version 5*. Rapport ISO TC184/SC4/WG5.
20. Lee, Y.C., Fu, K.S. (1987). Machine understanding of CSG : extraction and unification of manufacturing features. *IEEE Computer Graphics and Applications*, **7**(1), pp. 20-32.
21. Perng, D.B., Chen, Z., Li, R.K. (1990). Automatic 3D machining feature extraction from 3D CSG solid input. *Computer-Aided Design*, **22**(5), pp. 285-295.
22. Shpitalni, M., Fischer, A. (1994). Separation of disconnected machining regions on the basis of a CSG model. *Computer-Aided Design*, **26**(1), pp. 46-58.
23. Kyprianou, L.K. (1980). *Shape classification in computer-Aided design*. Thèse de doctorat, Christ College, Université de Cambridge.
24. Shah, J. (1991). Assessment of features technology. *Computer-Aided Design*, **23**(5), pp. 331-343.
25. Regli, W. (1992). *A survey of automated feature recognition techniques*. Rapport technique TR-92-18, Systems Research Center, Université du Maryland.
26. Gardan, Y., Minich, C. (1992). La modélisation géométrique et l'extraction de caractéristiques de forme. *Revue int. de CFAO et d'informatique graphique*, **7**(3), pp. 311-333.
27. Case, K., Gao, J. (1993). Feature technology : an overview. *Int. J. of Computer Integrated Manufacturing*, **6**(1/2), pp. 2-12.
28. Wu, M., Liu, C. (1996). Analysis on machined feature recognition techniques based on B-rep. *Computer-Aided Design*, **28**(8), pp. 603-616.
29. Minich, C. (1996). Un bilan des techniques d'extraction de caractéristiques de forme. *Revue int. de CFAO et d'informatique graphique*, **11**(6), pp. 591-615.
30. Ansaldi, S. (1985). Geometric modeling of solid objects by using a face adjacency graph representation. *ACM Computer Graphics*, **19**(3), pp. 131-139
31. de Floriani, L. (1987). A graph based approach to object feature recognition. *Proc. of the 3rd Symposium on Computational Geometry*, Waterloo, Ont., pp. 100-109
32. Falcidieno, B., Giannini, F. (1989). Automatic recognition and representation of shape-based features in a geometric modeling system. *Computer Vision, Graphics, and Image Processing*, **48**, pp. 93-123.
33. Lentz, D., Sowerby, R., (1993). Feature extraction of concave and convex regions and their intersections. *Computer-Aided Design*, **5**(7), pp. 421-437.
34. Chuang, S., Henderson, M. (1990). Three-dimensionnal shape pattern recognition using vertex classification and vertex-edge graphs. *Computer-Aided Design*, **22**(6), pp. 377-387.

35. Joshi, S., Chang, T. (1988). Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer-Aided Design*, **20**(2), pp. 58-66.
36. Sakurai, H., Gossard, D. (1990). Recognizing shape features in solid models. *IEEE Computer Graphics and Applications*, **10**(5), pp. 22-32.
37. Laakko, T., Mäntylä, M. (1993). Feature modelling by incremental feature recognition. *Computer-Aided Design*, **5**(8), pp. 479-492.
38. Marefat, M., Kashyap, R. (1990). Geometric reasoning for recognition of three-dimensionnal object features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **12**(10), pp. 949-965.
39. Gavankar, P., Henderson, M. (1990). Graph-based extraction of protrusions and depressions from boundary representations. *Computer-Aided Design*, **22**(7), pp. 442-450.
40. Dong, X., Wozny, M. (1991). A method for generating volumetric features from surface features. *Proc. of the Symposium on solid modeling foundations and CAD/CAM applications*, Austin, TX, pp. 185-194
41. Tseng, Y., Joshi, S. (1994). Recognizing multiple interpretations of interacting machining features. *Computer-Aided Design*, **26**(9), pp. 667-688.
42. Sapidis, N., Perucchio, R. (1992). Solid/solid classification operations for recursive spatial decomposition and domain triangulation of solid models. *Computer-Aided Design*, **24**(10), pp. 517-529.
43. Trabelsi, A., Meeran S., Carrard, M. (1997). A methodology for the generation of multiple interpretations of 2.5D spatially interacting features. *Revue int. de CFAO et d'informatique graphique*, **12**(3), pp. 257-276.
44. Woo, T. (1982). Feature extraction by volume decomposition. *Proc. of Conf. on CAD/CAM technology in mechanical engineering*, Cambridge, MA.
45. Kim, Y.S. (1992). Recognition of form features using convex decomposition. C*omputer-Aided Design*, **24**(9), pp. 461-476.
46. Parienté, F., Kim, Y.S. (1996). Incremental and localized update of convex decomposition used for form feature recognition. *Computer-Aided design*, **26**(8), pp. 589-602.
47. Waco, D., Kim, Y.S. (1994). Geometric reasoning for machining features using convex decomposition. *Computer-Aided design*, **26**(6), pp. 477-489.
48. Vandenbrande, J., Requicha, A. (1993). Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **15**(12), pp. 1269-1285.
49. Han, J., Requicha, A. (1997). Integration of feature based design and feature recognition. *Computer-Aided Design*, **29**(5), pp. 393-403.
50. Regli, W. (1995). Geometric algorithms for recognition of features from solid models. Thèse de doctorat, Université du Maryland.
51. Prabhakar, S., Henderson, M. (1992). Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *Computer-Aided Design*, **24**(7), pp. 381-393.
52. Nezis, K., Vosniakos, G. (1997). Recognizing 2½D shape features using a neural network and heuristics. *Computer-Aided Design*, **29**(7), pp. 523-539.

53. Chen, C.L.P, LeClair, S.R. (1994). Integration of design and manufacturing : solving setup generation and feature sequencing using an unsupervised-learning approach. *Computer-Aided Design*, **26**(1), pp. 59-75.
54. Wu, M.C., Chen, J.R., Jen, S.R. (1994). Global shape information modelling and classification of 2D workpieces. *Int. J. of Computer Integrated Manufacturing*, **7**(5), pp. 261-275.
55. Staley, S., Anderson, D., Henderson, M. (1983). Using syntactic pattern recognition to extract feature information from a solid geometric database. *Computers in Mechanical Engineering*, **2**, pp. 61-66.
56. Jakubowski, R. (1985). Extraction of shape features for syntactic recognition of mechanical parts. *IEEE Trans. on Systems, Man and Cybernetics*, **15**(5).
57. Choi, B., Barash, M., Anderson, D. (1984). Automatic recognition of machined surfaces from a 3D solid model. *Computer-Aided Design*, **16(**2), pp. 81-86.
58. Fu, Z., de Pennington, A., Saia, A. (1993). Graph grammar approach to feature representation and transformation. *Int. J. of Computer Integrated Manufacturing*, **6**(1/2), pp. 137-151.
59. Pinilla, J.M., Finger, S., Prinz, F.B. (1989). Shape feature description using an augmented topology graph grammar. *Proc. of NSF Engineering Design Research Conf*, Amherst, MA, pp. 285-300.
60. Spatial Technology Inc., (1998). *ACIS 3D toolkit : SAT format, version 4.0*. Spatial Technology Inc., Boulder, CO
61. Corney, J. (1997). 3D modeling with the ACIS kernel and toolkit. Wiley & Sons.
62. Corney, J., Clark, D. (1991). A feature recognition algorithm for multiply connected depressions and protrusions in 2½D objects. *Proc. of the Symposium on solid modeling foundations and CAD/CAM applications*, Austin, TX, pp. 171-183.

ANNEXE A

LISTE DES GÉONS DU PROJET

Les cinq attributs intrinsèques engendrent une possibilité de 72 géons, listés au tableau A.1. Le code ID correspond à l'étiquette du géon, alors que les codes MT, AT, ET, S, SF abrègent le nom d'attribut indiqué au tableau 3.1 (p. 71) :

- MT : matérialité (*Material Type*)
  - ⇒ +1 : positif
  - ⇒ −1 : négatif
- AT : courbure de l'axe (*Axis Type*)
  - ⇒ 1 : droit
  - ⇒ 2 : incurvé
- ET : courbure des arêtes génératrices (*Edge Type*)
  - ⇒ 1 : droites
  - ⇒ 2 : courbes
  - ⇒ 3 : hybrides
- S : symétrie du profil générateur (*Symmetry*)
  - ⇒ 1 : asymétrie
  - ⇒ 2 : symétrie réflective
  - ⇒ 3 : symétrie réflective et rotationnelle
- SF : fonction de balayage (*Sweep Function*)
  - ⇒ 1 : constante
  - ⇒ 2 : variable

Les géons utilisés dans ce projet sont indiqués dans le tableau A.1 par une trame de fond.

ID	MT	AT	ET	S	SF
1	+1	1	1	1	1
2	+1	1	1	1	2
3	+1	1	1	2	1
4	+1	1	1	2	2
5	+1	1	1	3	1
6	+1	1	1	3	2
7	+1	1	2	1	1
8	+1	1	2	1	2
9	+1	1	2	2	1
10	+1	1	2	2	2
11	+1	1	2	3	1
12	+1	1	2	3	2
13	+1	1	3	1	1
14	+1	1	3	1	2
15	+1	1	3	2	1
16	+1	1	3	2	2
17	+1	1	3	3	1
18	+1	1	3	3	2

ID	MT	AT	ET	S	SF
19	+1	2	1	1	1
20	+1	2	1	1	2
21	+1	2	1	2	1
22	+1	2	1	2	2
23	+1	2	1	3	1
24	+1	2	1	3	2
25	+1	2	2	1	1
26	+1	2	2	1	2
27	+1	2	2	2	1
28	+1	2	2	2	2
29	+1	2	2	3	1
30	+1	2	2	3	2
31	+1	2	3	1	1
32	+1	2	3	1	2
33	+1	2	3	2	1
34	+1	2	3	2	2
35	+1	2	3	3	1
36	+1	2	3	3	2

Tableau A.1 : Liste complète des combinaisons d'attribut (début)

ID	MT	AT	ET	S	SF
37	–1	1	1	1	1
38	–1	1	1	1	2
39	–1	1	1	2	1
40	–1	1	1	2	2
41	–1	1	1	3	1
42	–1	1	1	3	2
43	–1	1	2	1	1
44	–1	1	2	1	2
45	–1	1	2	2	1
46	–1	1	2	2	2
47	–1	1	2	3	1
48	–1	1	2	3	2
49	–1	1	3	1	1
50	–1	1	3	1	2
51	–1	1	3	2	1
52	–1	1	3	2	2
53	–1	1	3	3	1
54	–1	1	3	3	2

ID	MT	AT	ET	S	SF
55	–1	2	1	1	1
56	–1	2	1	1	2
57	–1	2	1	2	1
58	–1	2	1	2	2
59	–1	2	1	3	1
60	–1	2	1	3	2
61	–1	2	2	1	1
62	–1	2	2	1	2
63	–1	2	2	2	1
64	–1	2	2	2	2
65	–1	2	2	3	1
66	–1	2	2	3	2
67	–1	2	3	1	1
68	–1	2	3	1	2
69	–1	2	3	2	1
70	–1	2	3	2	2
71	–1	2	3	3	1
72	–1	2	3	3	2

Tableau A.1 : Liste complète des combinaisons d'attribut (fin)

ANNEXE B

DESCRIPTION DU FORMAT NEUTRE

Le format neutre de ProEngineer est un fichier ASCII qui décrit un modèle CAO selon le schéma de représentation Brep. De plus, il s'agit d'une forme énumérative F(E), où les entités sont décrites paramétriquement. Ce fichier est divisé en sept sections distinctes :

1. Informations générales
2. Dimensions
3. Caractéristiques de forme
4. Surfaces
5. Arêtes
6. Références
7. Unités de mesure

La description des surfaces et des arêtes sont les seules informations qui intéressent réellement notre projet. Avec l'évolution du projet, les données provenant d'autres sections du fichier pourraient cependant être utilisées en complément d'information pour les algorithmes. Compte tenu de nos besoins actuels, nous ne décrirons donc les sections 1, 2, 3, 6 et 7 que sommairement, pour nous attarder un peu plus sur les sections 4 et 5.

B.1 Sections d'intérêt moindre

La section "Informations générales" est assez courte, et précise surtout la valeur de certaines variables globales qui concernent l'objet entier : propriétés massiques, coefficients de réflection spéculaire et diffuse des surfaces, etc. Une seule donnée de cette section pourrait éventuellement servir dans le futur, soit la valeur de la variable *outline*, qui définit les deux coins opposés du parallélépipède qui contient la totalité de l'objet. Ce volume spatial définit virtuellement la dimension du matériau brut dans lequel est taillé l'objet ; plusieurs algorithmes amorcent effectivement leur raisonnement à partir de cette information (*root first*).

La section "Dimensions" est elle aussi assez brève, et précise les cotes et tolérances associées aux différentes parties de l'objet. Il s'agit surtout des dimensions que l'usager veut bien mettre en évidence sur la copie papier, puisque les dimensions de toutes les entités de l'objet doivent nécessairement être définies lors de la modélisation.

La troisième section, celle des "Caractéristiques de forme", est un peu différente de ce qui se fait en matière de descriptions CAO, dans la mesure où elle constitue l'effort de ProEngineer pour ajouter une information de plus haut niveau au modèle. Il s'agit en l'occurrence d'une énumération des différentes caractéristiques de forme présentes dans l'objet, ainsi que des surfaces qui leurs sont associées. Il existe environ une quinzaine de caractéristiques admises dans un fichier neutre, identifiées par des descripteurs comme "Protrusion", "Trou", "Coupe" (*cut*), "Arbre" (*shaft*), etc.

Bien que section attire l'attention dans un projet comme le nôtre, les caractéristiques de forme énumérées dans le fichier neutre souffrent d'un problème similaire à celui des représentations CGS : elles indiquent non pas les entités volumétriques de l'objet final, mais plutôt les *opérations* requises pour construire l'objet. Les descripteurs mentionnés plus haut correspondent en fait aux principales options de modélisation offertes par le logiciel. En deux mots, il n'y a aucune "intelligence" dans les caractéristiques de formes énumérées ; elles résument seulement la séquence de modélisation spécifique à l'utilisateur du logiciel. S'il est manifeste qu'un objet simple sera généralement modélisé de la même façon par des usagers différents, il en va tout autrement d'un objet complexe ; le nombre de descripteurs et de combinaisons possibles pour définir cet objet rendraient ardue toute inférence géométrique fondée sur ce type d'information. Nous ne faisons donc aucun usage des informations présentes dans cette section, bien qu'elles pourraient éventuellement servir pour la validation des résultats.

La sixième section, dite "Références" (*Datum*), définit toutes les entités qui ont servi à construire l'objet, mais qui n'en font pas partie : plans de références, axes de

cylindres, etc. La dernière section du fichier précise finalement si les dimensions de l'objet sont exprimées en pouces, en centimètres ou autre.

B.2 Description des surfaces

La section "Surfaces" décrit généralement la totalité des surfaces de l'objet. Il peut sembler paradoxal que certaines surfaces soient apparemment absentes de l'énumération : ceci dépend de la méthode de construction choisie par l'usager. Nous abordons ce problème au § 4.2.4.1. Remarquer que cette particularité n'est d'ailleurs pas exclusive au format neutre.

Le plus simple sera sans doute d'étudier un exemple de description de surface. Noter au demeurant que nous ne désirons pas faire part de tous les détails, mais seulement donner un aperçu d'une description typique. La section débute toujours par une étiquette du genre :

```
1 surfaces [14]
```

Le premier chiffre indique un début de section, suivi du descripteur nominatif de la section, et du nombre d'éléments décrits. Pour cet exemple, l'objet contenait donc 14 surfaces. Vient ensuite la description de chaque surface ; nous présentons ici l'une d'entre elles :

```
 1          2 surfaces
 2          3 id 22
 3          3 uv_min [2]
 4          4 uv_min 2*0.
 5          3 uv_max [2]
 6          4 uv_max 10.,3.
 7          3 xyz_min [3]
 8          4 xyz_min 3*0.
 9          3 xyz_max [3]
10          4 xyz_max 3.,10.,0.
```

```
11          3 orient 1
12          3 loops [2]
13          4 loops
14          5 edge_ids [6]
15          6 edge_ids 55,58,46,60,23,17
16          4 loops
17          5 edge_ids [2]
18          6 edge_ids 79,80
19          3 surface_type 34
20          3 surface(plane) ->
21          4 e1 [3]
22          5 e1 0.,1.,0.
23          4 e2 [3]
24          5 e2 1.,2*0.
25          4 e3 [3]
26          5 e3 2*0.,-1.
27          4 origin [3]
28          5 origin 3*0.
```

Pour la clarté de notre propos, nous avons ajouté un numéro de ligne, qui ne fait cependant pas partie du fichier neutre. Nous commentons ici les attributs en fonction de leur numéro de ligne :

- Ligne 2 : il s'agit de l'étiquette associée à la surface. Ces étiquettes sont attribuées séquentiellement par le logiciel, et chacune est unique dans tout le fichier, nonobstant le type d'entité.

- Lignes 3 à 6 : elles décrivent la plage de variation des paramètres u et v de la surface. Remarquer d'ailleurs la facheuse convention adoptée par ProEngineer quant à la répétition de nombres : le cas échéant, le caractère * indique une répétition de la valeur qui suit. Cette convention peut apparaître n'importe où dans le fichier. L'inconvénient de cette méthode est qu'à chaque opération de lecture du fichier, il devient impossible de saisir directement des valeurs numériques : il faut nécessai-

rement lire une chaîne de caractères, qui doit ensuite être analysée syntaxiquement dans l'éventualité d'un caractère de répétition.

- Lignes 7 à 10 : elles expriment les coordonnées cartésiennes des coins opposés d'un parallélépipède qui contient entièrement la surface.

- Ligne 11 : l'attribut d'orientation peut prendre deux valeurs, soit –1 et 1. Pour un plan, l'orientation indique si la normale, mentionnée plus loin, pointe vers l'extérieur ou vers l'intérieur de l'objet. Pour toutes les autres surfaces, forcément incurvées, ce drapeau remplace l'information fournie par la normale, et indique si le matériau est situé vers le centre de courbure de l'objet ou non.

- Ligne 12 : l'attribut `loops` indique le nombre de boucles de la surface, soit deux pour cet exemple. Tel que mentionné au paragraphe 1.5.3, toute surface présentera au moins une boucle ; il n'y a cependant pas de nombre maximal de boucles.

- Lignes 13 à 18 : elles explicitent les boucles une à une, en mentionnant l'étiquette de chaque arête de chaque boucle ; la description précise des arêtes apparaît dans la prochaine section du fichier. Détail important par contre, lorsqu'il y a plus d'une boucle, aucune convention ou identificateur n'indique quelle boucle est externe, et quelles autres sont internes.

- Lignes 19 et 20 : ces lignes précisent le type de surface, autant par un identificateur numérique que par un descripteur symbolique. Le nombre et la signification précise des attributs mentionnés par la suite dépendent bien sûr du type de surface décrite. Les surfaces admises par ProEngineer sont d'ailleurs nombreuses :

– surface plane	– surface de révolution	– surface de spline bicubique
– surface cylindrique	– surface réglée	– surface NURBS
– surface conique	– cylindre tabulé	
– surface torique	– surface de congé	

- Lignes 21 à 28 : cette partie de la description indique les vecteurs de paramétrisation de la surface, identiquement à ce qui est présenté à l'annexe C. Des attributs plus nombreux et de signification différente permettent de caractériser les autres types de surfaces.

B.3 Description des arêtes

Les arêtes sont décrites immédiatement après les surfaces ; cette section occupe d'ailleurs la plus grande partie du fichier, d'une part parce que les arêtes sont nécessairement plus nombreuses que les surfaces, et d'autre part parce que chaque arête est décrite sous deux formes paramétriques différentes.

Comme dans le cas précédent, la section s'amorce par le descripteur nominatif, suivi du nombre d'arêtes décrites :

```
1 edges [27]
```

Les arêtes sont ensuite décrites l'une après l'autre selon le modèle suivant :

```
1         2 edges
2         3 id 34
3         3 surface_ids [2]
4         4 surface_ids 10,45
5         3 direction [2]
6         4 direction 1,-1
7         3 uv_points [8]
8         4 uv_points 10.,3*0.,10.,2*1.,0.
9         3 curve_type 2
10        3 curve(line) ->
11        4 end1 [3]
12        5 end1 0.,10.,0.
13        4 end2 [3]
14        5 end2 0.,10.,1.
15        3 uv_curves[0] ->
16        4 curve_type 19
```

```
17          4 curve(b_spline) ->
18          5 degree 3
19          5 params [8]
20          6 params 4*0.,4*1.
21          5 c_pnts [12]
22          6 c_pnts 0.,2*0.,10.,0.333333333333333,0.,10.,
23          6 0.666666666666667,0.,10.,1.,0
24          5 weights [4]
25          6 weights 4*1.
26          4 flip 1
27          3 uv_curves[1] ->
28          4 curve_type 19
29          4 curve(b_spline) ->
30          5 degree 3
31          5 params [8]
32          6 params 4*0.,4*1.
33          5 c_pnts [12]
34          6 c_pnts 3*0.,0.333333333333333,2*0.,
35          6 0.666666666666667,2*0.,1.,2*0.
36          5 weights [4]
37          6 weights 4*1.
38          4 flip 1
```

Voici la signification des attributs d'arêtes, commentés ligne par ligne :

- Ligne 1 : début de description d'une arête.

- Ligne 2 : étiquette de l'arête. Noter au passage que la chronologie des descriptions d'arêtes est indépendante des étiquettes. Ceci peut rendre la lecture du fichier particulièrement longue lorsque le programme d'application recherche une arête spécifique, à moins de recourir à un système de repérage quelconque.

- Lignes 3 et 4 : étiquettes des surfaces adjacentes à chaque arête. Il y a toujours deux surfaces de part et d'autre de n'importe quelle arête.

- Lignes 5 et 6 : le drapeau `direction` indique si les extrémités de l'arête sont énumérées ou non dans le sens conventionnel de parcours des boucles. En effet, chaque arête doit appartenir à deux boucles, associées à chacune des surfaces voisines ; la convention de parcours des boucles impose alors que l'arête soit parcourue dans un sens pour une des boucles, et en sens inverse pour l'autre boucle. Parmi les deux extrémités d'arêtes mentionnées plus loin, ce drapeau précise donc quelles sont les coordonnées respectives de début et de fin d'arête, en fonction de la surface considérée. ProEngineer utilise la règle de la main gauche comme sens de parcours conventionnel.

- Lignes 7 et 8 : elles indiquent la position de l'arête dans l'espace paramétrique de chaque surface ; ces paramètres u et v ne doivent d'ailleurs pas être confondus avec la paramétrisation intrinsèque de l'arête, en t. L'objectif de cette énumération est de fournir à l'application qui exploite les données une localisation au moins approximative de l'arête par rapport à chaque surface parente. À cet effet, l'arête est échantillonnée en un certain nombre de points u et v sur chaque surface, et leurs valeurs respectives sont énumérées dans le fichier neutre selon la séquence suivante :

$$(u_1, v_1)_{\text{surface1}}\ ,\ (u_1, v_1)_{\text{surface2}}\ ,$$
$$(u_2, v_2)_{\text{surface1}}\ ,\ (u_2, v_2)_{\text{surface2}}\ ,$$
$$\text{etc}$$

La figure B.1 illustre par exemple le cas d'un arc de cercle, échantillonné à 5 endroits différents sur la courbe ; les valeurs de u et v mentionnées dans le fichier seraient celles qui correspondent à chacun de ces points, pour chacune des surfaces en cause. En pratique, le nombre de points d'échantillonnage varie selon le type et la dimension de l'arête : par exemple, un arc de cercle pourrait être échantillonné en une dizaine de points, alors qu'une ligne droite sera seulement échantillonnée à ses deux extrémités, comme c'est le cas dans l'exemple considéré.

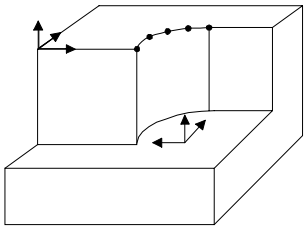


Figure B.1 : Échantillonnage paramétrique.

- Lignes 9 et 10 : identificateur numérique et descripteur symbolique du type de courbe. ProEngineer n'admet que quatre types d'arêtes, soit la ligne droite, l'arc de cercle, le spline cubique et le NURBS.

- Lignes 11 à 14 : description de l'arête dans le repère universel. Seules les coordonnées spatiales des extrémités de l'arête apparaissent ici, puisqu'il s'agit dans cet exemple d'une ligne droite. D'autres attributs spécifiques s'ajouteraient cependant pour décrire correctement les autres types d'arêtes.

- Lignes 15 à 38 : description de l'arête dans les repères respectifs de chaque surface latérale. Contrairement à la description précédente qui dépend du type d'arête, la description courante est toujours exprimée sous forme de NURBS ; pour les cas simples, les poids du NURBS sont unitaires, et la description devient alors celle d'un B-spline. L'attribut `params` correspond au vecteur de noeuds, et `c_pnts` aux points de contrôle. Noter que les coordonnées (u,v,w) de ces points de contrôle sont exprimées dans le système d'axe $e_1e_2e_3$ de chaque surface, plutôt que dans le système d'axe XYZ du repère universel.

ANNEXE C

FORMES MATHÉMATIQUES PARAMÉTRIQUES

Nous étudierons les représentations paramétriques de différents types d'entités concernées par ce projet. Notre objectif n'est pas de faire une étude approfondie du sujet, mais uniquement de rappeler les rudiments mathématiques requis pour manipuler certaines entités CAO. Nous profiterons de cette étude pour souligner certains modèles possibles de représentation de géons. Les noms des variables paramétriques sont ceux du format neutre lorsque l'entité est admise dans notre domaine d'entrée.

C.1 Entités de type arête

C.1.1 Ligne droite

La ligne droite est paramétrée sur un intervalle $0 \le t \le 1$, en fonction de deux vecteurs $\mathbf{end}_1$ et $\mathbf{end}_2$ qui expriment les coordonnées du début et de la fin de l'arête :

$$\mathbf{L} = (1 - t)\ \mathbf{end}_1 + t\ \mathbf{end}_2$$

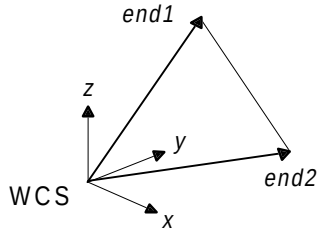


Figure C.1 : Paramétrisation de la ligne.

C.1.2 Arc de cercle

L'arc de cercle est paramétré sur un intervalle $0 \le t \le 1$, en fonction de deux vecteurs orthonormés $\mathbf{v}_1$ et $\mathbf{v}_2$ qui définissent le plan de l'arc. Si cet arc de rayon r est centré sur la coordonnée **o**, qu'il débute à l'angle θ_1 et se termine à l'angle θ_2 , alors :

$$\mathbf{A} = \mathbf{o} + r\left[\cos T\ \mathbf{v}_1 + \sin T\ \mathbf{v}_2\right]$$

avec $T = (1 - t)\ \theta_1 + t\ \theta_2$

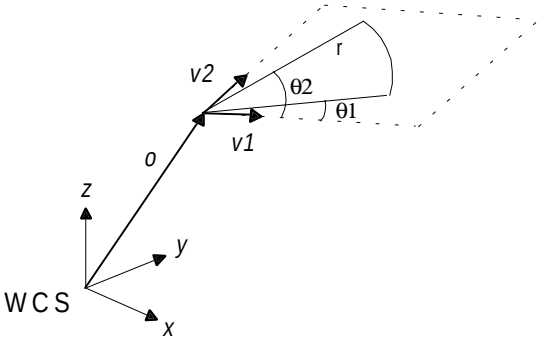


Figure C.2 : Paramétrisation de l'arc.

C.1.3 B-splines

Les splines sont des courbes paramétriques, approximées par une série de fonctions polynomiales ; elles ont l'avantage de permettre la modélisation de formes très diverses, pour un coût mathématique relativement faible. La forme générale des splines est :

$$\mathbf{P}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots \\ b_0 + b_1 t + b_2 t^2 + b_3 t^3 + \ldots \\ c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \ldots \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & \ldots \\ b_0 & b_1 & b_2 & b_3 & \ldots \\ c_0 & c_1 & c_2 & c_3 & \ldots \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \\ \ldots \end{bmatrix}$$

On retrouve dans les faits une grande variété de splines, ayant chacunes leurs particularités : splines cubiques, splines d'Hermite, splines de Bézier, β–splines, etc. Nous n'étudierons ici que le type de spline susceptible d'apparaître dans nos données, à savoir le B-spline.

Le B-spline est une courbe divisée en plusieurs intervalles, ayant chacun leur propre ensemble de fonctions polynomiales. De plus, un B-spline requiert des *points de contrôle*, c'est-à-dire des points (x,y,z) définis par l'utilisateur, et dont il tend à épouser le tracé. Le B-spline a pour équation générale :

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{X}_i B_{i,k}(t) \qquad \text{(éq. 1)}$$

avec $\mathbf{X}_i$: points de contrôle — $k-1$: degré du spline

$B_{i,k}$: fonctions de mélange — $n+1$: nombre de points de contrôle

t : paramètre

Dans l'équation précédente, k représente l'*ordre* du spline, choisi par l'utilisateur. Noter que dans la terminologie des B-splines, l'ordre est différent du *degré* du spline, qui est

égal à (k-1). Plus le degré du spline est grand, plus la courbe s'approche des points de contrôle.

Les *fonctions de mélange* $B_{i,k}$ (dites *blending functions* ou *basis functions*) sont en fait les polynômes en t, spécifiques à chaque intervalle i de la courbe ; un spline de degré k-1 sera justement approximé par des fonctions de mélange du même degré. Bien que les valeurs de n et k puissent être choisies indépendamment l'une de l'autre, le paramètre t doit être toujours positif. Ce paramètre s'étale par définition dans la plage :

$$0 \leq t \leq (n+1) - (k-1) \qquad \text{(éq. 2)}$$

Il en découle donc que les valeurs de n et k doivent respecter la contrainte suivante :

$$(n+1) - (k-1) > 0 \qquad \text{(éq. 3)}$$

Les fonctions de mélange sont calculées récursivement par la formule de Cox-deBoor :

$$B_{i,1} = \begin{cases} 1 & \text{si } v_i \leq t < v_{i+1} \\ 0 & \text{sinon} \end{cases} \qquad \text{(éq. 4)}$$

$$B_{i,k} = \frac{t - v_i}{v_{i+k-1} - v_i} B_{i,k-1} + \frac{v_{i+k} - t}{v_{i+k} - v_{i+1}} B_{i+1,k-1} \qquad \text{(éq. 5)}$$

Les v_i sont les *noeuds* du spline, c'est-à-dire les valeurs du paramètre t où s'effectue le découpage de la courbe. Le vecteur de noeuds est choisi par l'usager, en fonction du contrôle et des propriétés désirées sur la courbe ; le nombre de noeuds (et donc de sous-intervalles sur la courbe) est cependant invariable :

$$\| \mathbf{v} \| = n + k + 1 \qquad \text{(éq. 6)}$$

Le vecteur de noeud est toujours indexé de $0 \leq i \leq (\|\mathbf{v}\| - 1)$ pour la cohérence avec l'équation (1).

En pratique, trois types de noeuds sont possibles :

1. Les noeuds uniformes, qui sont régulièrement espacés le long de la courbe :

$$v_i = i \times \frac{(n+1)-(k-1)}{\|\mathbf{v}\|} \qquad \text{(éq. 7a)}$$

2. Les noeuds uniformes ouverts[18], pour lesquels il y a k répétitions des valeurs de noeuds à chaque extrémité du vecteur :

$$v_i = \begin{cases} 0 & \text{si } i < k \\ i-k+1 & \text{si } k \le i \le n \\ n-k+2 & \text{si } i > n \end{cases} \qquad \text{(éq. 7b)}$$

3. Et les noeuds non-uniformes, qui doivent seulement répondre à la relation :

$$v_i \le v_{i+1} \qquad \text{(éq. 7c)}$$

Pour fixer les idées, calculons par exemple un B-spline quadratique défini par 4 points de contrôle ; on a alors k = 3 et n = 3, ce qui respecte la contrainte de l'équation (3).

1. Déterminer la plage de variation du paramètre t :

Par l'équation (2), $0 \le t \le (3+1)-(3-1)$

donc $0 \le t \le 2$

2. Déterminer la taille du vecteur de noeuds :

Par l'équation (6), $\|\mathbf{v}\| = 3+3+1 = 7$

3. Déterminer la valeur des noeuds :

Nous choisissons arbitrairement un vecteur de noeuds uniformes ouverts. Par l'éq. (7b),

$$\mathbf{v} = [0\ 0\ 0\ 1\ 2\ 2\ 2]$$

Remarquer que dans les calculs qui suivent, **v** sera indexé de v_0 à v_6.

4. Calculer les fonctions de mélange :

L'examen de l'équation (1) indique qu'il faudra connaître les valeurs de $B_{0,3}$, $B_{1,3}$, $B_{2,3}$ et $B_{3,3}$; en effet :

$$\mathbf{P}(t) = \sum_{i=0}^{3} \mathbf{X}_i B_{i,3}(t) = \mathbf{X}_0 B_{0,3} + \mathbf{X}_1 B_{1,3} + \mathbf{X}_2 B_{2,3} + \mathbf{X}_3 B_{3,3}$$

En appliquant l'équation (5), on obtient donc :

$$B_{0,3} = \frac{t - v_0}{v_2 - v_0} B_{0,2} + \frac{v_3 - t}{v_3 - v_1} B_{1,2} = (1 - t) B_{1,2}$$

$$B_{1,3} = \frac{t - v_1}{v_3 - v_1} B_{1,2} + \frac{v_4 - t}{v_4 - v_2} B_{2,2} = t B_{1,2} + \tfrac{1}{2}(2 - t) B_{2,2}$$

$$B_{2,3} = \frac{t - v_2}{v_4 - v_2} B_{2,2} + \frac{v_5 - t}{v_5 - v_3} B_{3,2} = \tfrac{1}{2} t B_{2,2} + (2 - t) B_{3,2}$$

$$B_{3,3} = \frac{t - v_3}{v_5 - v_3} B_{3,2} + \frac{v_6 - t}{v_6 - v_4} B_{4,2} = (t - 1) B_{3,2}$$

Par définition, on pose égal à 0 tout terme dont le dénominateur est nul ; ce cas se produit par exemple dans le calcul de $B_{0,3}$ et $B_{3,3}$. Les fonctions de mélange $B_{i,3}$ calculées requièrent à leur tour de déterminer $B_{1,2}$, $B_{2,2}$ et $B_{3,2}$; en procédant similairement par l'équation (5), avec k = 2, on trouve :

---

[18] Certains auteurs les qualifient simplement de noeuds non-uniformes.

$$B_{1,2} = \frac{t - v_1}{v_2 - v_1} B_{1,1} + \frac{v_3 - t}{v_3 - v_2} B_{2,1} = (1 - t)\, B_{2,1}$$

$$B_{2,2} = \frac{t - v_2}{v_3 - v_2} B_{2,1} + \frac{v_4 - t}{v_4 - v_3} B_{3,1} = t\, B_{2,1} + (2 - t)\, B_{3,1}$$

$$B_{3,2} = \frac{t - v_3}{v_4 - v_3} B_{3,1} + \frac{v_5 - t}{v_5 - v_4} B_{4,1} = (t - 1)\, B_{3,1}$$

Reste donc à calculer $B_{2,1}$ et $B_{3,1}$; par l'équation (4) :

$$B_{2,1} = \begin{cases} 1 & \text{si } 0 \le t < 1 \\ 0 & \text{sinon} \end{cases} \qquad B_{3,1} = \begin{cases} 1 & \text{si } 1 \le t < 2 \\ 0 & \text{sinon} \end{cases}$$

5. Substituer les termes dans l'équation du spline :

Selon l'équation du spline :

$$\mathbf{P}(t) = \sum_{i=0}^{3} \mathbf{X}_i B_{i,3}(t) = \mathbf{X}_0 B_{0,3} + \mathbf{X}_1 B_{1,3} + \mathbf{X}_2 B_{2,3} + \mathbf{X}_3 B_{3,3}$$

Après substitution et simplification, le spline se résume donc à :

$$\begin{aligned} \mathbf{P}(t) = {} & \mathbf{X}_0 \left[(1 - 2t + t^2)\, B_{2,1} \right] + \\ & \mathbf{X}_1 \left[(2t - \tfrac{3}{2} t^2)\, B_{2,1} + (2 - 2t + \tfrac{1}{2} t^2)\, B_{3,1} \right] + \\ & \mathbf{X}_2 \left[(\tfrac{1}{2} t^2)\, B_{2,1} + (-2 + 4t - \tfrac{3}{2} t^2)\, B_{3,1} \right] + \\ & \mathbf{X}_3 \left[(1 - 2t + t^2)\, B_{3,1} \right] \end{aligned}$$

On constate effectivement que l'approximation polynomiale est de degré 2, tel que désiré. Pour l'intérêt de la chose, nous avons tracé ce spline pour deux ensembles de points de contrôle. On observe sur la figure C.3 que la courbe est toujours définie par deux paraboles jointes bout-à-bout.

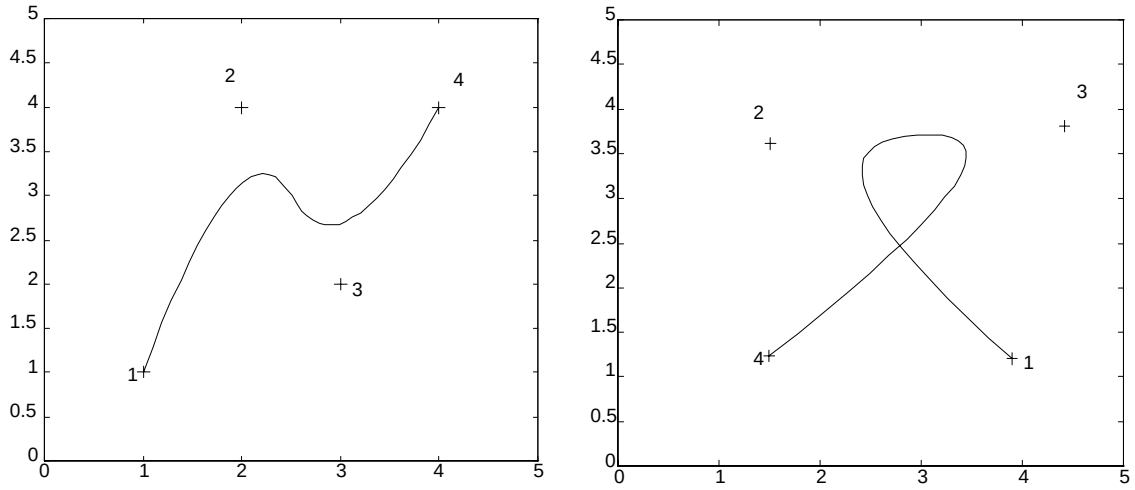


Figure C.3 : Splines quadratiques.

Nous terminerons cette étude des B-splines en mentionnant en vrac quelques unes de leurs propriétés :

- Si le B-spline s'approche plus ou moins des points de contrôle qui le définissent, il passe toujours par contre par le premier et le dernier point de contrôle. Cette propriété permet donc de produire une courbe fermée en posant comme dernier point de contrôle la valeur du premier.

- Un B-spline est toujours entièrement délimité par la coquille convexe des points de contrôle ;

- Un avantage des B-splines sur d'autres types de splines est la possibilité de contrôler localement la courbe. En effet, pour un spline d'ordre k, le déplacement d'un point de contrôle affecte k intervalles de la courbe ; ceci contraste avec d'autres types de splines, où le déplacement d'un seul point de contrôle modifie la totalité de la courbe.

- La *continuité* de la courbe sur chaque intervalle du paramètre dépend uniquement de l'ordre k du spline. La continuité réfère aux dérivées de la courbe :

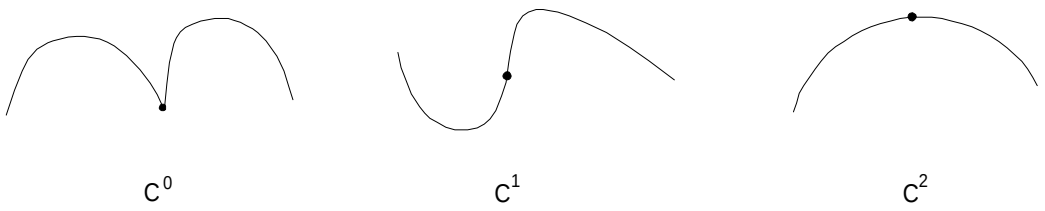


Fig. C.4 : Continuité des splines.

- une continuité d'ordre 0 (C^0) indique que les différents segments de la courbe sont seulement liés à leurs points de jonction, c'est-à-dire aux noeuds.

- pour une continuité d'ordre 1 (C^1), les dérivées paramétriques premières de deux sections successives sont égales à leur point de jonction, c'est-à-dire qu'elles partagent la même tangente à cet endroit.

- pour une continuité d'ordre 2 (C^2), les dérivées premières et secondes de deux sections successives sont égales à leur point de jonction : elles ont donc en plus le même type de courbure au noeud.

On démontre qu'un spline de degré k-1 possède une continuité C^{k-2} ; par exemple, on observe clairement à la figure C.3 que les splines de degré 2 ont seulement une continuité de dérivée première. Noter d'autre part qu'un spline de degré 1 aurait une continuité C^0 : en l'occurrence, il s'agirait d'une courbe approximée par des segments de droite, reliant chacun des points de contrôle.

Ce lien entre l'ordre du spline et sa continuité explique d'ailleurs pourquoi les splines de degré 3 (k = 4) sont si souvent utilisés en modélisation : ce sont des entités qui présentent un bon compromis entre le coût de calcul et le niveau d'approximation de la courbe. De plus, le degré 3 est la valeur minimale pour éviter une inversion intempestive de courbure entre différents intervalles du spline.

- Toute transformation affine sur le spline s'effectue directement sur les points de contrôle : l'avantage en coût de calcul est très évident, puisqu'il n'y a aucun besoin de redéfinir le spline à chaque transformation, et que le nombre de points de contrôle demeure généralement faible.

- Rappelons finalement qu'il existe plusieurs façons de contrôler la forme d'un B-spline :

  – par le degré polynomial du spline ;
  – par le nombre de points de contrôle ;
  – par la position des points de contrôle ;
  – par la répétition de certains points de contrôle ;
  – par la valeur des poids ;
  – et par le type de vecteur de noeuds.

C.1.4 NURBS

Les NURBS (*Non-Uniform Rational B-Splines*) sont en fait une généralisation des B-splines, qui permettent de modéliser exactement les coniques :

$$\mathbf{P}(t) = \frac{\sum_{i=0}^{n} \mathbf{X}_i w_i B_{i,k}(t)}{\sum_{i=0}^{n} w_i B_{i,k}(t)}$$

Compte tenu de l'ampleur de la présentation précédente sur les splines, d'une part, et de notre rejet des entités sculptées pour ce projet, d'autre part, nous n'entrerons pas dans le détail des NURBS. On doit souligner au passage que si les NURBS sont *a priori* des courbes, un maillage de NURBS définit alors une surface du même nom. L'équation précédente est alors généralisée à deux paramètres u et v.

C.2 Entités de type surface

C.2.1 Surface plane

Tout point d'un plan peut être situé à partir de trois vecteurs **o**, $\mathbf{e}_1$ et $\mathbf{e}_2$ selon la relation :

$$\mathbf{P} = \mathbf{o} + u\mathbf{e}_1 + v\mathbf{e}_2$$

Le vecteur **o** situe l'origine du plan par rapport au repère universel WCS, alors que $\mathbf{e}_1$ et $\mathbf{e}_2$ sont deux vecteurs orthogonaux et unitaires, parallèles au plan. Le vecteur $\mathbf{e}_3$ représente la normale au plan. Noter par ailleurs que l'origine du plan peut se situer à l'extérieur de la surface plane, mais toujours dans son prolongement.

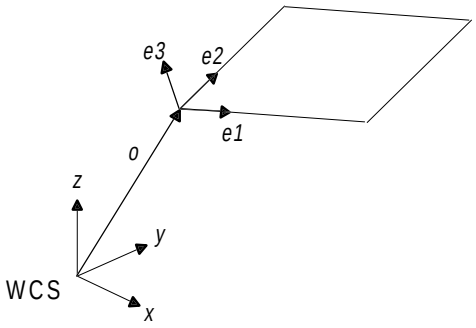


Fig. C.5 : Paramétrisation du plan.

C.2.2 Surface cylindrique

Une surface cylindrique de rayon r est paramétrée en fonction de trois vecteurs orthogonaux et unitaires $\mathbf{e}_1$, $\mathbf{e}_2$ et $\mathbf{e}_3$, et d'un vecteur **o** qui situe l'origine du cylindre par rapport au repère universel :

$$\mathbf{P} = \mathbf{o} + r\left[\cos u\ \mathbf{e}_1 + \sin u\ \mathbf{e}_2\right] + v\mathbf{e}_3$$

Comme pour le plan, l'origine de la surface peut se situer à l'extérieur du cylindre, mais toujours dans son axe. Noter que la hauteur du cylindre est connue par la valeur limite du paramètre v.

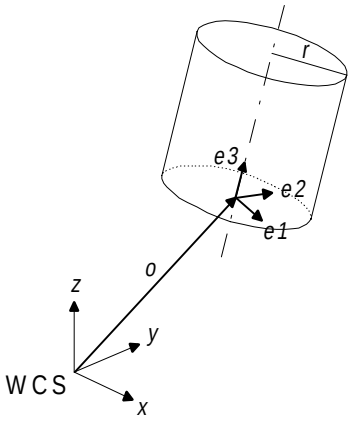


Fig. C.6 : Paramétrisation du cylindre.

C.2.3 Surface conique

Une surface conique ouverte d'un angle α est paramétrée en fonction de trois vecteurs orthogonaux et unitaires $\mathbf{e}_1$, $\mathbf{e}_2$ et $\mathbf{e}_3$, et d'un vecteur **o** qui situe l'origine du cône par rapport au repère universel :

$$\mathbf{P} = \mathbf{o} + v \tan\alpha \left[\cos u\, \mathbf{e}_1 + \sin u\, \mathbf{e}_2\right] + v\mathbf{e}_3$$

La distance radiale d'un point de la surface est de $r = v \tan\alpha$. L'angle d'ouverture du cône est compris entre $-\pi/2 < \alpha < \pi/2$.

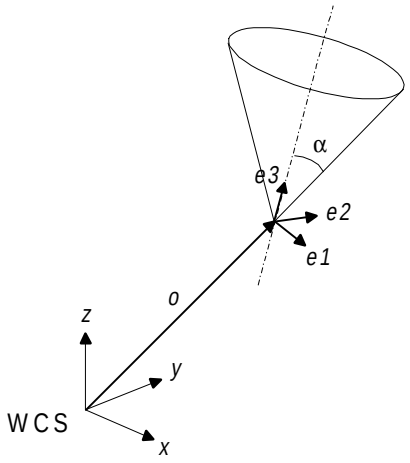


Fig. C.7 : Paramétrisation du cône.

C.2.4 Superquadriques

Les superquadriques sont des entités superficielles qui proviennent de la généralisation de de l'équation de l'ellipsoïde[19] :

$$\left[\left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon_2}}\right]^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon_1}} = 1$$

En effet, on reconnaît aisément l'ellipsoïde lorsque $\varepsilon_1 = \varepsilon_2 = 1$. La forme implicite de l'équation précédente la rend cependant difficile à manipuler ; les superquadriques sont donc toujours mathématisées sous la forme paramétrique :

---

[19] À cet égard, il y a un abus de langage généralisé dans la littérature : les dites superquadriques sont en fait des *superellipsoïdes*. Ces dernières sont un sous-ensemble des superquadriques.

$$\mathbf{X} = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \begin{bmatrix} a_1 (\cos(u))^{\varepsilon_1} (\cos(v))^{\varepsilon_2} \\ a_2 (\cos(u))^{\varepsilon_1} (\sin(v))^{\varepsilon_2} \\ a_3 (\sin(u))^{\varepsilon_1} \end{bmatrix} \qquad \text{avec} \begin{cases} -\frac{\pi}{2} \le u \le \frac{\pi}{2} \\ -\pi \le v \le \pi \end{cases}$$

Dans un référentiel sphérique centré sur l'objet, les paramètres u et v expriment respectivement la latitude et la longitude à la surface de la superquadrique, alors que ε_1 et ε_2 définissent la "rondeur" de l'objet dans chacune de ces directions. Tel que l'illustre la fig. C.8, les scalaires a_1, a_2 et a_3 sont de simples facteurs d'échelle selon les axes x, y et z.

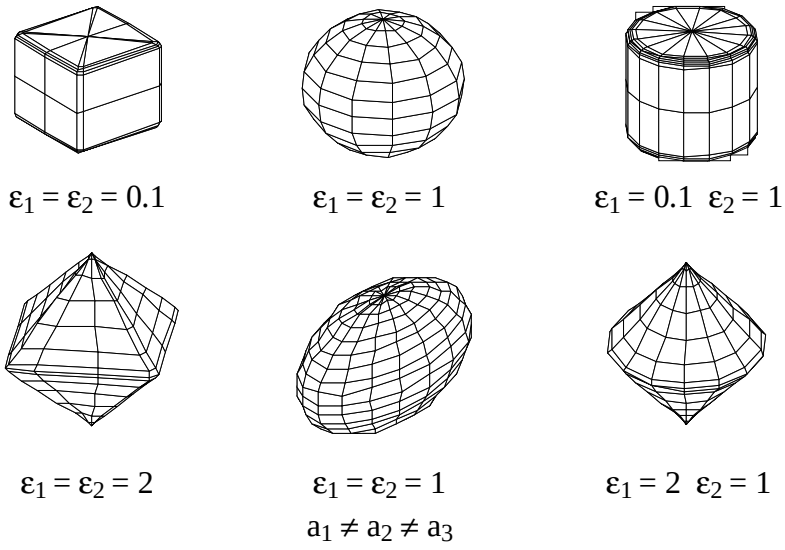


Figure C.8 : Superquadriques.

Les superquadriques sont aussi sujettes aux opérations de torsion, pliage et fuselage. Par exemple, la transformation suivante appliquée sur une superquadrique [x y z] permet d'engendrer une superquadrique [X Y Z] incurvée d'une courbure $\kappa = 1/r$ dans le plan xz :

$$\begin{cases} X = \dfrac{1}{\kappa} - \left(\dfrac{1}{\kappa} - x\right)\cos(\kappa z) \\ Y = y \\ Z = \left(\dfrac{1}{\kappa} - x\right)\sin(\kappa z) \end{cases}$$

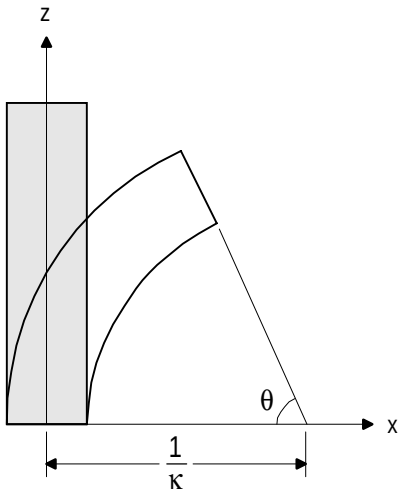


Fig. C.9 : Superquadrique courbée.
(Adapté de [13])

Dans une perspective de géonisation d'objets, il est pertinent de rappeler qu'une superquadrique définit toujours une surface complètement fermée, tel qu'illustré à la figure C.8. Il y aurait donc un certain intérêt à représenter les corps par superquadriques, puisque celles-ci permettent de modéliser un géon complet avec une seule équation. Il serait par contre hors de question de modéliser un objet complexe avec une seule superquadrique, puisque ce type de modèle mathématique n'autorise aucun contrôle local sur la forme de la surface. En pratique, un objet formé de plusieurs géons devrait être représenté par autant de superquadriques.

Les superquadriques imposent aussi des contraintes de symétrie sur les géons. En effet, toute superquadrique possède un plan et un axe de symétrie, l'un perpendiculaire à l'autre ; ceci restreint notoirement la portée des superquadriques pour la géonisation d'objets manufacturés. On notera d'ailleurs que toute superquadrique est un cylindre généralisé, mais que n'importe quel cylindre généralisé n'est pas forcément une superquadrique.

Ces différentes contraintes rendent inappropriées les superquadriques pour ce projet, bien qu'elles ont souvent été utilisées en vision pour ajuster des données à un modèle (*data fitting*).

C.2.5 Hyperquadriques

Nous mentionnons finalement les hyperquadriques pour tenter de faire le tour des représentations mathématiques ; il faut cependant souligner que ces entités sont d'un usage très marginal, et n'apparaissent à toutes fins pratiques jamais dans une description CAO. Comme les superquadriques, les hyperquadriques définissent un volume fermé en une seule équation ; elles permettent par contre un contrôle local de l'enveloppe, et n'imposent aucune contrainte de symétrie. En principe, un objet complet peut donc être modélisé avec une seule hyperquadrique, quelle que soit sa complexité géométrique :

$$\sum_{i=1}^{N} \left| H_i(x, y, z) \right|^{\gamma_i} + \sum_{j=1}^{M} c_j e^{-\sum_{p=1}^{P_i} \left| K_{jp}(x,y,z) \right|^{\gamma_{jp}}}$$

Nous ne croyons pas que ce type de représentation soit vraiment pertinent pour un projet comme le nôtre, surtout parce qu'il est très difficile à manipuler. En effet, il n'existe pas de paramétrisation connue pour l'hyperquadrique, et la forme algébrique implicite de l'équation rend tout calcul très laborieux.