Stellingen behorend bij het proefschrift getiteld :

Automated Fabrication of Shape Models of Free-form Objects with a Sculpturing Robot

Johan W.H. Tangelder

1. De rekentijd voor 6-assig frezen kan aanzienlijk gereduceerd worden door het freesprobleem te splitsen in een bereikbaarheid- en een padplanningdeelprobleem.
2. Snelle fabricage van vormmodellen van dubbelgekromde objecten vereist een afweging van de tijd om het freesproces uit te voeren en het berekenen daarvan.
3. De formele beschrijving van volumeverwijdering en robot padplanning door Minkowski-operaties biedt de mogelijkheid om methoden uit de beeldbewerking en de computationele geometrie toe te passen om het freesprocess te optimaliseren.
4. Voor een snelle geautomatiseerde fabricage van grote vormmodellen ($>$ 1 meter doorsnede) met een gangbare industriële robot, is een combinatie van object decompositie, materiaal-verwijdering en hercompositie het meest geschikt.
5. Zowel bij een robot die een gereedschap langs korte lijnstukjes beweegt, als bij een auto die de maximum snelheid in acht houdt, is snelle acceleratie en deacceleratie belangrijker dan snelheid.
6. Aangezien technologie zich moet richten op de behoeften van mensen, is de inbreng van de subfaculteit Industrieel Ontwerpen in het multidicsiplinaire onderzoek van de Delftse Interfacultaire OnderzoekCentra (DIOC's) onontbeerlijk.
7. Als we n-dimensional dammen als volgt beschrijven:
   Ieder van de 10^n velden van het dambord wordt gerepresenteerd door een vector $(i_1, i_2, \ldots i_n)$ met $1 \leq i_j \leq 10$ voor $j = 1, \ldots n$. Een zet van veld v naar veld w is alleen geoorloofd als $v - w = (\pm 1, \pm 1, \ldots \pm 1)$. Het speelveld van het dambord bestaat uit de velden die door een schijf bereikt kunnen worden vanuit veld $(1, 1, \ldots 1)$ door het doen van geoorloofde zetten.
   Dan wordt slechts het 2^{n-1}-ste deel van het dambord bespeeld.
8. In de parabel van de verloren zoon (Lucas 15:11-32) kan de liefdevolle vader, die zijn jongste zoon laat vertrekken naar een ver land, geïdentificeerd worden met een God die de menselijke keuzevrijheid respecteert. Een God die alles heeft voorbeschikt en gepland, respecteert de menselijke keuzevrijheid niet en zou meer lijken op een supercomputer dan op een liefdevolle vader.
9. Met behulp van robottechnologie zal in de volgende eeuw opnieuw in Nederland steenkool gewonnen worden.
10. Aangezien in Delft kinderen jonger dan vier jaar niet worden toegelaten tot de promotiezitting, verdient het aanbeveling om gelijktijdig "een uurtje voor jonge onderzoekers" te houden.

Theses belonging to the thesis entitled :

Automated Fabrication of Shape Models of Free-form Objects with a Sculpturing Robot

Johan W.H. Tangelder

1. The computation time of sculpturing with 6-axis milling can be significantly reduced by decomposing the sculpturing problem into an accessibility subproblem and a path planning subproblem.
2. Fast fabrication of shape models of free-form objects requires a trade-off in terms of the time to carry out the milling process and the time to compute it.
3. The formal description of volume removal and robot path planning by Minkowski operations offers the opportunity to apply image processing and computational geometry methods to optimize the computation of the milling process.
4. For fast automated fabrication of big (> 1 metre in diameter) shape models with a common industrial robot, a combination of object decomposition, material removal and recomposition is the most appropriate.
5. Both for a robot that moves a tool along short line segments, and for a car keeping a speed limit, high acceleration and deceleration are more important than the speed.
6. Since technology should address the needs of people, it is indispensable that the Subfaculty of Industrial Design Engineering contributes to to the multi-disciplinary research carried out by the Delft Interfaculty Research Centers (DIOC's).
7. If we describe n-dimensional draughts as follows:
   Each of the 10^n squares of the draughtboard is represented by a vector $(i_1, i_2, \ldots i_n)$ with $1 \leq i_j \leq 10$ for $j = 1, \ldots n$. A move from square v to a square w is allowed only if $v - w = (\pm 1, \pm 1, \ldots \pm 1)$. The playing-field of the draughtboard consists of the squares that can be reached from the square $(1, 1, \ldots 1)$ by allowed moves.
   Then only a (2^{n-1})-th fraction of all squares is used as playing field.
8. In the parable of the lost son (Luke 15:11-32) the loving father, who allows his youngest son to set off for a distant country, can be identified with a God respecting the human freedom of choice. A God who preordains and plans everything, does not respect the human freedom of choice and would resemble a supercomputer instead of a loving father.
9. Coal mining will be reintroduced in the Netherlands in the next century, based on robot technology.
10. Since in Delft children under four years of age are not allowed at the thesis defense, an "hour for young researchers" should be organized.

Automated Fabrication of Shape Models of Free-form Objects with a Sculpturing Robot

Automated Fabrication of Shape Models of Free-form Objects with a Sculpturing Robot

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College voor Promoties aangewezen,
op dinsdag 8 september 1998 te 10.30 uur
door

Johan Wilhelmus Hendrik TANGELDER

wiskundig ingenieur
geboren te Brunssum

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr. I. Horváth
Prof.dr. M.H. Overmars

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr. I. Horváth,	Technische Universiteit Delft, promotor
Prof.dr. M.H. Overmars,	Universiteit Utrecht, promotor
Prof.ir. P. de Ruwe,	Technische Universiteit Delft
Prof.dr. I.T. Young,	Technische Universiteit Delft
Prof.dr. F. Kimura,	Universiteit van Tokio
Prof.dr. T.C. Woo,	Universiteit van Washington
Dr. J.S.M. Vergeest,	Technische Universiteit Delft

ISBN 90-9011783-0

Printed in the Netherlands

voor Helma en onze kinderen

Preface

This thesis has resulted from the research project "Rapid Prototyping using Virtual Reality and Robot Milling", which started in 1994 at the Faculty of Industrial Design Engineering of Delft University of Technology. The project aimed at the development of both virtual and physical shape modeling techniques to support conceptual shape design. The thesis focuses on the development of shape sculpturing as an advanced physical shape modeling technique using robot milling. The technique has been implemented, evaluated and applied on the Sculpturing Robot system at Delft University of Technology.

Chapter 1 is a general introduction into the different disciplines the research is related to, namely industrial design engineering, physical shape modeling, and tool path generation for sculptured surface milling.

Chapter 2 describes shape sculpturing as a robot motion planning problem. We discuss the capabilities of the sculpturing process with regard to the shapes that can be machined exactly as well as the complexity of the sculpturing process. We propose a decomposition of the sculpturing problem into the subproblem of extracting a limited number of tool access orientations from the geometry of a free-form object, and the subproblem of generating the milling paths given any fixed tool access orientation.

Chapter 3 presents a method to compute a limited number of tool access orientations by accessibility analysis of the free-form object.

Chapter 4 shows that for a fixed tool orientation, spatial planning and Minkowski operations can be applied to state the sculpturing problem, and to describe, implement and compare NC machining methods.

Chapter 5 describes the Sculpturing Robot (SR) system and the Multiple Access Orientation Sculptor (MAOS) software package to implement the methods from Chapter 3 and 4 with the SR system.

Chapter 6 evaluates the methods both from Chapter 3 and 4 separately, as well as the sculpturing strategy implemented by the software package.

Chapter 7 provides conclusions drawn based upon the research described in this thesis.

Many of the results in this thesis have been published. The paper (Tangelder, Vergeest & Overmars 1996) describes the method to extract tool access orientations from the geometry of a free-form object. The papers (Tangelder, Vergeest & Overmars 1997*a*) and (Tangelder, Vergeest & Overmars 1998) discuss the application of spatial planning and Minkowski operation to NC machining. Finally, the paper (Tangelder, Vergeest, van den Belt & Overmars 1997*b*) summarizes this Ph.D. thesis.

Summary

A physical shape model provides better feedback to the designer than the visual presentation of a shape on a 3D CAD system. For automated fabrication of physical shape models NC milling is the main competitor to systems based on incremental methods. Completely automated NC milling has a significant impact on fast automated fabrication of shape models. Compared to 3-axis milling, 6-axis milling extends the class of shapes that can be fabricated automatically. Therefore, this thesis focuses on supporting automated fabrication of shape models by 6-axis milling.

Sculpturing shapes from a stock of material is a task that involves six degrees of freedom: the location as well as the orientation of the milling tool is allowed to change at any time in the milling process. To reduce the complexity of the sculpturing process the computation of the tool orientations and the tool locations are separated. First, a limited number of tool orientations is extracted from the geometry of the shape to be sculptured. Next, with each of these tool orientations, one or more milling stages are performed to obtain the shape.

The main results of the research project are

- A new method to compute a limited number of tool access orientations by accessibility analysis of a free-form object has been developed and implemented.
- A new formal method to describe tool path generation, taking into account interference avoidance for both the tool and the tool holder, has been developed.
- A methodology to apply 6-axis milling to the automated fabrication of shape models of free-form objects with a sculpturing robot, has been developed. This new approach has been successfully implemented in the MAOS software package and applied to fabricate medium sized shape models.

A series of experiments has been carried out to evaluate the MAOS software using three free-form objects that represent typical design engineering products with increasing complexity and with a different morphological character. The evaluation has shown that MAOS extends the shape domain of manufacturable objects significantly.

J.W.H. Tangelder, Delft University of Technology

Samenvatting

Tijdens het ontwerpproces is tussentijdse evaluatie van het ontwerp nodig. Een fysiek vormmodel geeft een betere terugkoppeling aan de ontwerper dan een visuele presentatie van een vorm op een 3D CAD systeem. Voor de automatische fabricage van fysieke vormmodellen is NC frezen het belangrijkste alternatief voor systemen die gebaseerd zijn op incrementele methoden. Volledig geautomatiseerd NC frezen levert een belangrijke bijdrage aan snelle fabricage van vormmodellen. Vergeleken met 3-assig frezen breidt 6-assig frezen de klasse van vormen die automatisch gemaakt kunnen worden uit. Daarom richt dit proefschrift zich op de ondersteuning van de automatische fabricage van vormmodellen door 6-assig frezen.

Het frezen van vormen uit een blok materiaal is een taak die 6 vrijheidsgraden heeft: zowel de lokatie als de oriëntatie van de frees kan veranderen op elk tijdstip in het freesproces. Om de complexiteit van het freesproces te reduceren zijn de berekening van de oriëntaties en de posities van de frees gescheiden. Eerst wordt uit de geometrie van de uit te frezen vorm een beperkt aantal oriëntaties van de frees berekend. Dan wordt voor ieder van deze oriëntaties een of meer freesfasen uitgevoerd om de vorm te verkrijgen.

De belangrijkste resultaten van dit ondezoeksproject zijn

- Gebaseerd op bereikbaarheidsanalyse is een nieuwe methode ontwikkeld om een beperkt aantal oriëntaties te berekenen om een frees te positioneren.
- Voor de beschrijving van freesbaangeneratie, waarbij zowel de botsing van de frees als van de freeshouder wordt vermeden, is een nieuwe formele methode ontwikkeld.
- Een methodologie voor het 6-assig frezen van vormmodellen van dubbelgekromde objecten met een freesrobot is ontwikkeld. Deze nieuwe aanpak is succesvol geïmplementeerd in het MAOS software pakket en toegepast om middelgrote vormmodellen te fabriceren.

Een aantal experimenten is uitgevoerd om de MAOS software te evalueren voor drie dubbelgekromde objecten, typisch voor het industrieel ontwerpen. Deze objecten hebben een toenemende complexiteit en een verschillend morfologisch karakter. De evaluatie heeft aangetoond dat MAOS de klasse van vormen van maakbare objecten aanzienlijk uitbreidt.

J.W.H. Tangelder, Technische Universiteit Delft

Acknowledgments

This Ph.D. research is part of the project "Rapid Prototyping using Virtual Reality and Robot Milling". I would like to thank Prof.dr. Gerda Smets, Prof.ir. Paul de Ruwe, Dr. Cees Overbeeke, Dr. Pieter Jan Stappers and Dr. Joris Vergeest for their effort in writing the research proposal for this project. I would also like to thank the Delft University Research Fund for supporting the project.

This Ph.D. research would not have been possible without the contribution and encouragement of a large number of people.

First of all, I would like to thank the people who were willing to join the committee for the defense of this Ph.D. thesis. Prof.ir. Paul de Ruwe was my advisor from the start of the project in December 1993. My promotor Prof.dr. Mark Overmars from Utrecht University joined the committee at the end of 1994 and traveled often to Delft. My promotor Prof.dr Imre Horváth joined the committee as well as the Subfaculty of Industrial Design Engineering in the spring of 1997. Joris Vergeest, my daily supervisor, was always willing to discuss my research and gave me many helpful suggestions. Each of these people have been a great help throughout the project, providing interesting opinions and comments to both the research and the thesis writing. Furthermore, I thank Prof. Ted Young for his valuable comments on my draft thesis. Special thanks must go to both Prof. Fumihiko Kimura and Prof. Tony Woo who had to travel a long distance, to be present in Delft at the defense of the thesis. They also provided helpful suggestions on my draft thesis.

Further, I would like to thank Ronald Tangelder for providing valuable comments on my draft thesis.

Also, I thank Bram de Smit, Han Broek, Adrie Kooijman and Henk van den Belt for their contributions in realizing the Sculpturing Robot system.

Finally, I would like to thank my other colleagues from the TPI and ICA projects, who gave me a pleasant environment in which I have worked for the last nine years.

Hans Tangelder

Contents

1

Introduction

1.1 Industrial design engineering and physical shape modeling

The historical development of engineering disciplines shows a diversification, from military to civil engineering, and then branching out towards separate fields like infrastructure, manufacturing, mining, naval and aviation technology. But the systematic development of mass-manufactured consumer durables and professional equipment which meet the needs of people, requires the integration of knowledge from several disciplines. This has, only recently led to the rise of the new discipline of *industrial design engineering* that integrates many aspects of the product development process: management of product development, product innovation, marketing, human factors, ergonomics, engineering, form-giving and aesthetics, and so on. Industrial design engineering research has to address the trends in the industry that change the product development process, like: global manufacturing, smarter products and systems, shorter time-to-market, and the demand for environmental preservation.

Many authors have published books about methods for product development and have presented the concept of the design cycle as a formalized description of the product development process. In this section we focus only on the first phase of the design process, the conceptual design phase. For a description of the complete design process the reader should consult one of the books by (Ullman 1992, Cross 1994, Baxter 1995, Roozenburg & Eekels 1995, Ulrich & Eppinger 1995, Pahl & Beitz 1996, Pugh 1996).

In this initial phase of the design process, a number of design concepts arise in the mind of the designer(s). During this phase product ideas are conceptualized and compared. This conceptual design phase actually deals with much more than just the shape of a product, but in this thesis only conceptual shape design will be addressed. Most designers draw their 3D shape concepts using 2D sketches on paper. Looking back at a sketch can inspire a different solution that would not have come to mind if the idea had not been drawn. It is well-know that the physical activity of sketching helps in generating new ideas

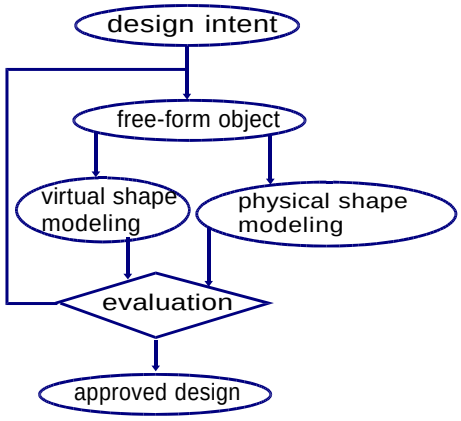


Figure 1.1 Supporting conceptual shape design with virtual and physical shape modeling.

(Goldschmidt 1991). With sketching the designer can generate and compare a number of 3D shape concepts quickly. Using traditional CAD systems it takes many hours or even days of concentrated effort to create 3D free-form CAD models for new designs. However, from the designer's point of view the ease of use of an advanced CAD system and sketching on paper by simple pen strokes should be comparable. In the past years, the development of powerful computing hardware led to an increasing number of sophisticated engineering applications. Especially in the area of mechanical systems design an immense progress in the development of modeling, analysis, and simulation software has been achieved. The new generation of 3D CAD systems provides high-level modeling operations in combination with virtual modeling techniques, including kinematic simulation, finite-element analysis, and NC path generation and simulation. Also, much research is devoted to computer support of the conceptual design phase (van Dijk 1994, Chu, Dani & Gadh 1997, Dani & Gadh 1997, Eggli, Hsu, Brüderlin & Elber 1997, Rix & Kress 1997, Tovey 1997, van Elsas 1997, Hummels, Smets & Overbeeke 1998, Hummels & Stappers 1998). With these new techniques the designer can quickly create a number of free-form objects and use virtual modeling to make a first comparison.

This thesis, however, focuses on the fast and inexpensive fabrication of physical shape models of free-form objects. Both virtual and physical shape modeling techniques support conceptual shape design as depicted in Figure 1.1. However, physical shape models provide feedback to the designer going beyond visual presentation. This additional feedback from physical shape models is especially relevant for

- **Evaluation**
  In practice designers find a graphical presentation of a design concept inefficient for a complete evaluation. Not only by looking at a physical model from all sides, but also by feeling and stroking using the fingertips, a number of improvements (Lennings 1997) can often be made.

- **Verification**
  With CAD images of complex shapes it is still very difficult to visualize exactly what the actual complex shape will look like. Some errors may still escape from the review of engineers and designers. Therefore, a thorough final check of the design using a physical model should not be omitted.

- **Communication**
  For the presentation of the design concept at meetings with customers, managers, and other members of the product development team shape models are indispensable.

- **Manufacturing**
  By providing a physical product at an earlier design state, we can speed up process planning and tooling design. In addition the model can help reduce problems in interpreting the blue prints on the shop floor.

- **Marketing**
  To assist product sales, a model can be used to demonstrate the concept, design ideas, as well as the company's ability to produce it. Also, the model can be used to gain feedback of customers for design modifications so that the final product will meet the requirements of the customers.

The outline of the remainder of this introduction is as follows. In Section 1.2 physical shape modeling techniques including 5-axis NC milling are discussed and compared. In Section 1.3 we state the requirements on advanced milling methods for physical shape modeling. The state-of-the-art in 5-axis tool path generation for sculptured surface milling is discussed in Section 1.4. The research issues that are addressed in this thesis are described in Section 1.5.

1.2 Comparison of methods for physical shape modeling

In this section we provide an overview of manufacturing processes and we compare their capabilities with respect to physical shape modeling. The books by (Kalpakjian 1992, Magrab 1997) provide an extensive description of manufacturing technology by casting, forming, welding and material-removal processes. The paper by (Kruth 1995) focuses on physical shape modeling processes by:

- **deforming methods**
  Deforming methods start from the right amount of bulk material and deform it to the right shape. Deforming then refers to either "deforming in solid state" (forging, stamping, drawing, extruding, etc.) or "deforming in liquid or semi-liquid state" (casting, injection moulding, etc.).

- **decremental methods**
  Decremental methods start from a larger amount of bulk material and gradually remove all excess material (turning, milling, grinding, etc.).

- **incremental methods**
  Incremental methods gradually add material until the required shape is created. Incremental machining includes welding as well as methods based upon layer building (selective laser sintering, laminated object manufacturing, fused deposition modeling, etc.).

(Kruth 1995) distinguishes further between traditional and non-traditional machining processes. Kruth defines *non-traditional machining* as recently developed processes that apply machining principles that differ substantially from those applied in traditional machining. Non-traditional deforming basically applies laser beam energy to deform material by inducing thermal stresses. (Thomson & Pridham 1997) have described a controlled laser forming process combined with laser cutting to support rapid shape modeling of sheet-type components like car door panels. Examples of non-traditional decremental methods are water jet machining, laser beam machining, and electrical discharge machining.

A lot of non-traditional machining processes based on incremental methods have been developed recently. For a more in-depth description of incremental methods the reader is referred to the book by (Beaman, Barlow, Bourell, Crawford, Marcus & McAlea 1997) and for a discussion about the open issues in incremental manufacturing the reader is referred to the Ph.D. thesis by (de Jager 1998) and the article by (Yan & Gu 1996).

Table 1.1 (adapted from (Kruth 1995)) presents an overview of manufacturing processes.

The needs for fast physical shape modeling can be very different and they obviously depend on the intended purpose (evaluation, presentation) of a shape model, and on its size. The requirements differ with regard to geometrical accuracy, material and mechanical properties, appearance, required dimensions, the number of copies to make, the cost of producing them, and the time this takes. A comparison of four different physical shape modeling technologies has been made by (Wall, Ulrich & Flowers 1992). They evaluated 5-axis NC machining, stereolithography and rubber moulding as physical shape modeling processes, and also included CAD solid modeling as a virtual shape modeling means. They concluded that depending on the specific situation each of these methods may be preferable.

(Yan & Gu 1996) review the main incremental physical shape modeling technologies. They identified as major problems in these technologies: part accuracy, limited material variety and mechanical performance. They present a practical application of their own Cubital Solider 4600 system: the fabrication of a master pattern to produce casting moulds of poly-urethane bicycle tyres. The new incremental physical shape modeling systems might be ideal in their ease of use, they are not suitable for fast concept evaluation because of the delays involved. In (Lennings 1997) the coffee break requirement is stated: Evaluation models are needed during the design process, and should be ready within

Type of Process	Traditional Manufacturing	Non-Traditional Manufacturing
Deforming methods	Forging Stamping Rubber molding Deep-drawing Bending Extruding Die Casting Rolling	Laser Forming
Decremental methods	3-axis NC for CAM 5-axis NC for CAM Turning Sawing Grinding	3-axis NC for physical shape modeling 6-axis NC for physical shape modeling LBM: Laser Beam Machining EDM: Electro-Discharge Machining WJM: Water Jet Machining
Incremental methods		SLS: Selective Laser Sintering STL: Stereolithography LOM: Laminated Object Manufacturing BPM: Ballistic Particle Manufacturing

Table 1.1 Overview of manufacturing processes (not exhaustive).

a coffee break, in order to continue the design process. The designer needs a personal desktop modeling system, with which he can produce, if needed, even a sequence of several shape concept models within one day. (Kleiman, de Jager & Lennings 1997) describe a number of these personal desktop modeling systems.

In their research book on incremental physical shape modeling (Beaman et al. 1997) state that NC milling is the main competitor to systems based on incremental methods: In comparison with these methods milling can produce parts with superior accuracy and surface finish; in addition, milling processes can operate on a much broader range of materials. Completely automated NC milling would have a significant impact on fast physical shape modeling. Current NC systems, however, are not generally considered to be fast shape modeling technologies because:

- they still require skillful human intervention to help plan the operations and to operate the equipment.
- custom fixturing and special tooling is often required.
- milling has inherent geometric limitations.

From the users point of view, to be acceptable as an aid in the conceptual design phase, it should be possible to invoke the physical shape modeling system without assistance of an expert in NC milling. This implies that the NC preparation methods normally offered on traditional CAD/CAM systems are not adequate. These methods may demand explicit subdivision of the geometric model into regions (e.g. regions corresponding to surface patches created by the designer to build the model), each of which must be separately associated with machining parameters. Collision checks are either absent or can be

Requirement	Expensive incremental methods	Inexpensive incremental methods	Expensive NC milling	Inexpensive NC milling
Shape domain	No limitations	No limitations	No internal structures	Part visible from one direction
Size of models	< 50x50x50 cm	< 50x20x20 cm	< 100x100x100 cm	< 10x10x10 cm
Turn around time	High	Low	High	Low
Accuracy	High	Low	High	Low
Use of models	Various	Concept model	Various	Concept model
Cost of labor	operator	-	CNC specialist	-
Cost of machine	> $ 100,000	< $ 100,000	> $ 100,000	< $ 100,000
Choice of materials	limited	limited	wide range	wide range

Table 1.2 Comparison of existing physical shape modeling technologies.

laborious derived by graphical means. Tool path verification and avoidance of machining clashes are for the largest part or totally the responsibility of the CAD user. Details such as approach tactics, milling tool selection and precision characteristics must be explicitly input. For a designer, who does not want to be a CAM specialist, a different type of CAM software is needed that is easy-to-use, and needs only a small number of milling parameters that have a default value that is suitable for fast shape modeling. (Lennings 1997) describes an example of this different approach to NC milling with an inexpensive modeling system consisting of a low-end 3-axis machine for milling foam models with a software package supporting user friendly NC.

Apart from user interface aspects, technical issues play a crucial role. One of them is the turn-around time for model production. Both the computation and the physical machining process must be sufficiently fast as to make creation of several alternative shapes within a few hours realistic. Note that this speed requirement conflicts with the demand of full automation.

There are, on the other hand, also NC requirements which can be relaxed for shape milling. In the conceptual design phase the gross outline of geometry is of main importance, allowing milling with moderate accuracy. Higher precision milling will be necessary not until the milled object receives the status of mockup. For shape checks easy-to-mill materials such as poly-urethane foam can be applied, so that no large tool force variations occur and feed rate control may be limited to setting an absolute maximum speed.

Table 1.2 presents a comparison of the physical shape modeling technologies that we have discussed above. In this table deforming methods have not be considered, because the mould which is used to manufacture one or more models, has to be manufactured first using another physical shape modeling technology. Therefore, with deforming methods small pilot series of a product can be manufactured fast, but they are not applied to produce only one shape model.

From Table 1.2 we conclude that the development of advanced milling methods for shape modeling extends the class of mid-size and large complex free-form shapes models that can be fabricated. Therefore, in the next section we will state requirements for the development of this technology to support conceptual design.

1.3 Requirements on advanced milling methods for physical shape modeling

Based on our discussion of the support to conceptual design by physical shape modeling and our comparison of existing physical shape modeling technologies, we state the following requirements on advanced milling methods for physical shape modeling:

1. The system must be safe. No injury to personnel, or damage to devices or workpieces is allowed to occur.

2. There must be no limit on the complexity of the input free-form object. Any shape that could be realized by some tool at some location with some orientation must be actually manufacturable. However, it would be acceptable that some portion of a shape cannot be realized due to physical inaccessibility of it by any tool.

3. The input data may be as inaccurate as the user-selected geometric accuracy, or resolution, of the physical shape model. This is an essential requirement since the system must be applicable for conceptual, initial design, where the product geometry is possibly only roughly specified, and is perhaps even incomplete. The system should adequately process such initial product geometry.

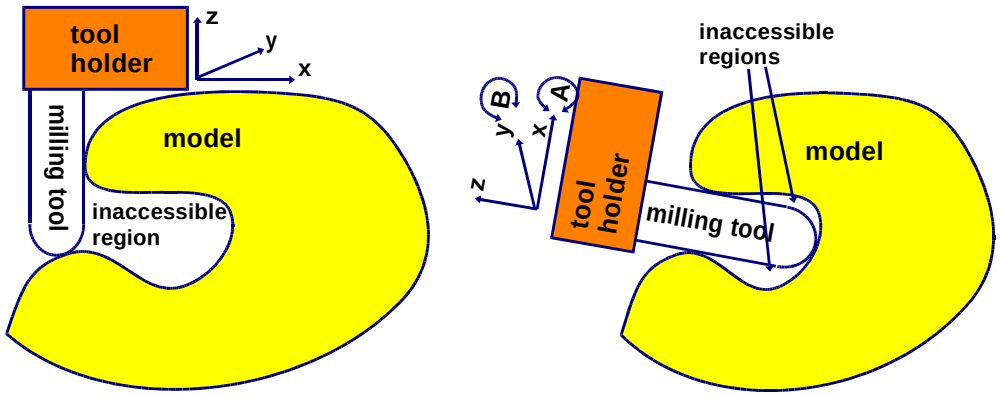
4. The process of data generation must be completely automatic. The user should not be concerned with any technical problem that occurs or could occur during machining, such as collisions or instability of the part-in-progress. The only selection the designer needs to make is the geometric accuracy and possibly a scaling factor.

5. The system must be able to manufacture models with dimensions up to $1 \times 1 \times 1$ metre without the need of refixturing the workpiece.

6. Depending on the application, the size, and the complexity of the models, a turn-around time between half an hour (coffee break requirement) and a few days is required.

7. Each milling process must be performed in such a way that a subsequent, more accurate machining process can be made. This implies that at no time material is removed from the final model, not even a fraction of the current accuracy.

Requirement 2 on the complexity of the shape model cannot be satisfied by a 3-axis NC milling machine. Requirement 5 on the size of the models cannot be satisfied by a desk top milling machine. Hence, at least the capabilities of a 5-axis NC milling machine are needed for shape modeling of mid-range and large complex free-from objects. Therefore, we review in the next section the state-of-the-art in 5-axis NC tool path generation.



(a) 3-axis milling. (b) 5-axis milling.

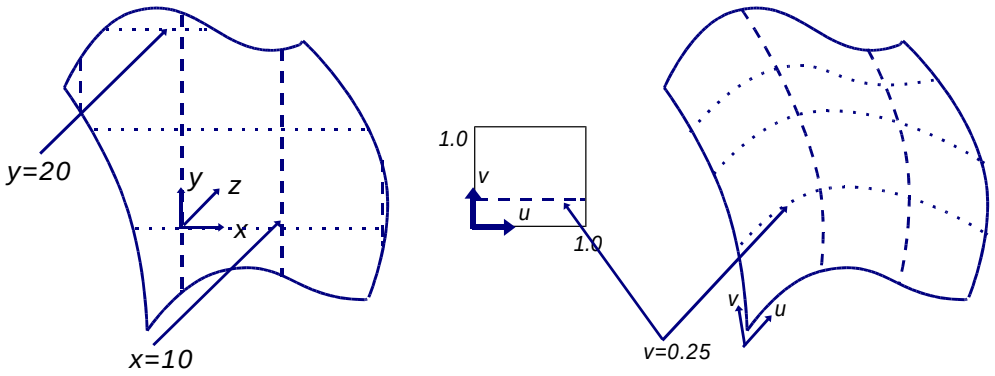
Figure 1.2 Comparison of 3-axis and 5-axis milling.

1.4 State-of-the-art in 5-axis NC tool path generation for sculptured surface milling

Several methods are in use in which free-form shape models are realized using milling machines. In 3-axis C(NC)[1] milling a tool is positioned with three degrees of freedom, *i.e.* a 3-axis NC milling machine can move a milling tool with a fixed orientation to any point in its workspace (See Figure 1.2(a)). This means that only those parts of a model can be milled that are visible from a particular direction. Inaccessible regions (under cuts) cannot be removed from the model. One way to generalize 3-axis milling is to add rotational degrees of freedom to the tool. The tool can now not only be positioned at a specific location, but can also be arbitrarily oriented. This milling mode is know as 5-axis milling, because of the five degrees of freedom provided. In general the use of a 5-axis milling machine results in much smaller inaccessible regions and thus in a better shape model (See Figure 1.2(b)).

Often the milling process is split into a roughing and a finishing stage. To produce a physical shape model efficiently from a stock of material, the bulk waste material is removed during the roughing stage by a cutting tool with a large radius. The stock is shrunk to a rough model. In the finishing stage the rough model is shrunk to the physical shape model by a cutting tool with a smaller radius. In this stage the cutting tool may follow a surface of the model. In most CAD systems the model surfaces are represented by a B-spline representation. For an introduction on B-spline representations the reader

[1]During the last years, the concept of 'numerical control' has been gradually replaced by a concept making more directly use of a computer: 'computerized numerical control' (CNC). In this thesis, we do not distinguish between them. Rather, we use the term 'NC' synonymously for any of the two concept.



(a) Cartesian paths. (b) isoparametric paths.

Figure 1.3 Comparison of Cartesian paths and isoparametric paths.

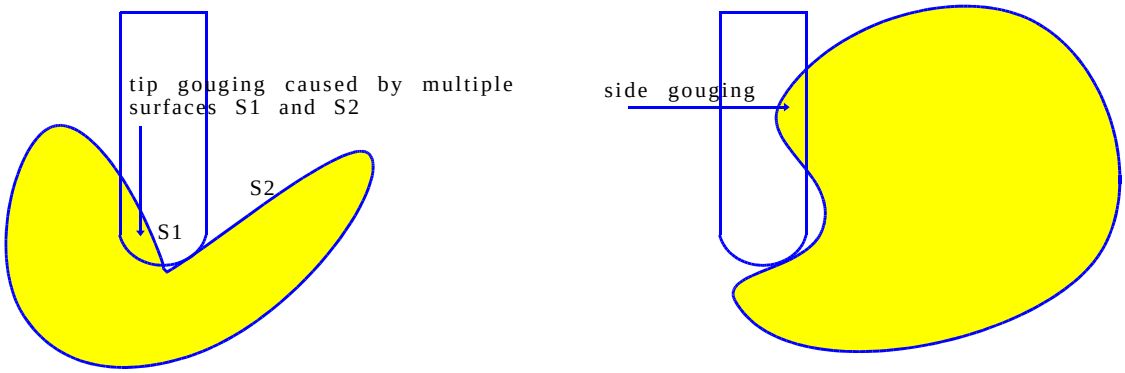
is referred to (Farin 1996).

The tool path pattern is critical to the efficiency of the milling process. In general, methods for machining free-form shapes generate zigzag paths. As illustrated by Figure 1.3(a) a zigzag path is either formed from parallel sweeps in Cartesian space or from sweeps across surfaces that are parallel parametrically. In the former case the paths are called *Cartesian* and in the latter case the paths are called *isoparametric*.

Since the late 1980's there has been an enormous amount of work published on NC tool path generation. We refer the reader to a number of recent surveys and reviews on NC tool path literature and we describe a number of interesting new papers that have been published recently.

(Dragomatz & Mann 1997) have provided an extensive bibliography of the literature on NC tool path generation. They have grouped 220 papers into the following categories: surveys, issues, systems, isoparametric paths, Cartesian paths, planar pocketing paths, sculptured surface pocketing paths, roughing paths, tool positioning, offset surface method, 5-axis methods, mesh models, pixel and point models, simulation and verification, space-filling curve based tool paths, cleanup cut tool paths, point-based roughing paths, and region decomposition. (Jensen & Anderson 1996) present a mathematical review of methods and algorithms used to compute milling cutter placement for multi-axis finished-surface milling. The survey by (Marshall & Griffiths 1994) focuses on path construction as classified by pocketing, surface-at-a-time, and whole model methods. (Choi, Chung & Park 1995) provide a comprehensive overview of the application of the Z-map model to various fields including tool path generation. A Z-map or height grid is an approximation of a surface by a number of height values at grid elements on the xy-plane that are stored as a 2D array $Z[i, j]$.

In recent years a lot of papers have been published on issues in path generation for



(a) Tip gouging at the boundary of multiple surfaces.

(b) Side gouging.

Figure 1.4 Examples of gouging.

5-axis NC milling. In the remainder of this section we classify a number of these papers according to the main aspect they cover.

- **Systems**
  (Ko, Kim, Park & Kim 1994) describe a method to sculpture models of human faces and its implementation on a 6-axis robot. (Tangelder & Vergeest 1994) describe a milling method for physical shape modeling and its implementation on a seven-degree-of-freedom sculpturing robot system consisting of a 6-axis robot and a turn table. By this method a shape model is milled with five fixed tool orientations perpendicular either to the upper face or to any of the four side faces of the initial stock containing the shape model.

- **Tool interference avoidance using accessibility analysis**
  A point of a shape is accessible by a milling tool if the tool can be positioned so that it touches the point and does not intersect the surface. Hence, by determining accessibility of a surface part, tool interference can be avoided and tool access directions can be found with which the surface part can be machined. (Elber 1994) presents an approach that reduces the accessibility problem of 5-axis milling using a flat end tool to a 3-axis accessibility problem. For a shape whose boundary is represented by B-spline surfaces, (Lee & Chang 1995) apply the convex hull property for B-spline surfaces to find a conservative approximation of the allowed tool access orientations.

- **Tool interference avoidance per tool position**
  Since an NC program for sculptured surface machining consists of a finite sequence of interpolated tool positions, a number of methods are based on the modification

or computation of individual tool positions. In sculptured surface machining interference between the tool and the surface to be machined is a serious problem. In the literature this kind of interference is called *gouging*. Gouging is often encountered when the tool size is too large relatively to the concave radius of curvature or if models are machined that are composed of multiple surfaces (see Figure 1.4). (Li & Jerard 1994) describe research on algorithms for the generation of gouge-free, Cartesian 5-axis tool paths across composite surface patches. In (Elber & Cohen 1996) the use of a symbolic approach to detect tool gouging in 5-axis milling is investigated and a heuristic approach to eliminate gouging is presented. (Lee 1997) proposes a method to find the allowed tool orientations by considering both local and global surface shapes. Based on the evaluation of local surface shape, a geometric analysis method is developed to first find a feasible tool orientation for gouging avoidance. Adjacent geometry is then taken into consideration for detecting possible side gouging. (Morishige, Kase & Takeuchi 1997) present a method that selects a feasible tool orientation from a 2-dimensional map representing the relationship between all tool attitudes and the existence of collision. (Cho, Lee & Kim 1997) present an algorithm for generating collision-free tool paths for 5-axis milling using the potential energy method. By virtually charging the tool and the surface to be machined with static electricity, global tool collision as well as local interference is eliminated. Moreover, machining efficiency is simultaneously improved by minimizing the curvature difference between the machined surface and the tool swept surface. In all these works it is assumed that the tool-end is used to obtain a shape, *i.e.* the tool-end is positioned on the surface to be milled. It is also possible to obtain a shape with the tool side. (Liu 1995) describes tool interference avoidance for this 5-axis side milling technique.

- **Accuracy**
  If paths are generated in a zigzag fashion scallops (ridges) will be left between adjacent tool paths. (Choi, Park & Jun 1993) present a method that minimizes scallop height for 5-axis milling by the selection of an optimal tool orientation per tool position.

- **Set-up orientations**
  In 5-axis machining the unit sphere is used to represent the directions with which the milling tool can be positioned. Due to limitations on the motion ranges of the rotational axes of a 5-axis milling machine, the directions with which a tool can be positioned are limited to a proper subset of the unit sphere (Kang & Suh 1997). Therefore, it is often needed to set-up the stock several times. The time to dismount, recalibrate, and remount the stock can be considerable in comparison to the actual milling time. Unfortunately, the problem of computing the minimal number of set-ups is NP-hard[2], *i.e.* probably it is not possible to solve this problem within a polynomial running time. So attention has focused on obtaining efficient algorithms that approximate closely the minimum number of set-ups. (Gupta, Janardan, Majhi

[2]NP is an abbrevation for Nondeterministic Polynomial.

& Woo 1996) apply computational geometry algorithms to approximate the minimal number of set-ups within a logarithmic factor. (Kang & Suh 1997) describe algorithms, that given a 5-axis machine and a surface, determine the machinability of the surface and a set-up giving the smallest rotational range of the tool.

- **Free-form shape analysis**
  Saddle-like and concave regions of a shape can be milled using a ball-end tool, but convex regions can be machined faster and with a smaller scallop using a flat-end tool. Therefore, (Elber 1995) applies curvature analysis to dichotomize a free-form surface into a region that can be milled with a ball-end tool and a region that can be milled with a flat-end tool.

- **Tool path verification**
  (You & Chu 1997) present a scheme to automatically detect tool interference for tool paths that have been generated for 5-axis machining of sculptured surfaces.

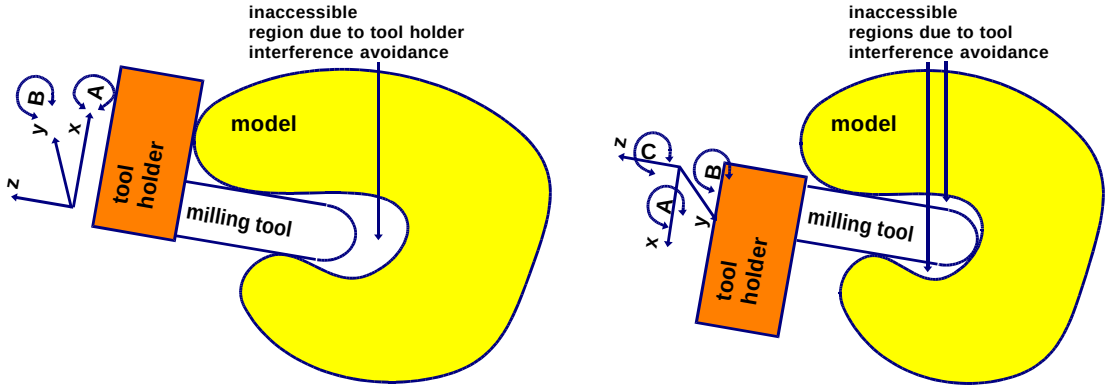
Most of the research that we have reviewed in this section, deals with tool interference avoidance. But for machining models of mid-range and large sizes, it is necessary to take also tool holder collision avoidance into account. In these cases the machining path contains often more than 10^5 tool positions. Therefore, it is not straightforward to develop efficient methods, that take, for each individual tool position, also tool holder interference avoidance into account. The methods that use accessibility analysis to avoid tool collision are more promising, but accessibility analysis for the tool only, does not avoid tool holder collision. Hence, new research should address the need for efficient collision-free machining.

Furthermore, reduction of the number of set-ups to machine a model would save time to dismount and remount the stock. Adding a sixth rotational axis to a milling device eliminates the need to use more than one set-up. With such a device it is possible to position the tool with every direction. Hence, all material outside the shape model can be reached. It is only necessary to keep a support structure for the shape model. Only, if one wants to machine away this support structure an extra set-up is needed.

Also, with a 6-axis machining device the tool holder can rotate around the milling tool. Figure 1.5 shows that with help of this extra rotational degree of freedom interference between the tool holder and the stock-in-progress, can often be avoided. Therefore, adding a sixth rotational axis to a 5-axis milling machine improves the collision avoidance capabilities of that machine further. However, to the best knowledge of the author, in the literature there is little attention given to the topic of 6-axis milling.

1.5 Scope of the research

Since 6-axis milling extends the capabilities of 5-axis milling, this thesis addresses the following research question: how can we support physical shape modeling by 6-axis milling processes? From the requirements on advanced milling methods for physical shape modeling from Section 1.3 we derive the following research issues:



(a) 5-axis milling. (b) 6-axis milling.

Figure 1.5 Comparison of 5-axis and 6-axis milling. The sixth axis of the milling machine can be used to avoid collision of the tool holder.

- **Interference avoidance between stock-in-progress and the tool holder**
  To meet requirement 1 and 4, collision avoidance between the tool holder and the stock-in-progress is indispensable. In the literature about 5-axis NC tool path generation, little consideration is paid to this topic.

- **Interference avoidance between the shape model and the milling tool**
  To meet requirement 7 the milling tool should never interfere with the shape model.

- **Extension of the shape domain of manufacturable objects**
  To meet requirement 2, advanced machining algorithms should make full use of the capabilities of 6-axis milling.

- **Fabrication of shape models of initial free-form objects**
  In conceptual design the product geometry may be only roughly specified. The software has to be able to process such initial free-form objects in order to meet requirement 3.

- **Machining large models**
  In order to meet the requirements 5 and 6, both the computation of the machining process and the machining process itself should be fast.

Since the time available for a Ph.D. project is limited we could not take into account some other issues. These include

- **Cutting off large chunks of foam**
  Faster manufacturing might be possible by cutting off large chunks of foam, e.g. with a hot wire or a hot knife.

- **Selecting milling tools by analysis of the geometry of the free-form object.**
  Given the geometry of an initial free-form object, the result of the machining process can be improved by selecting one or more milling tools. In our approach it is assumed that the milling tool is given beforehand.

- **Deviations of positioning and orientation of the milling device**
  We neglect the errors due to the limited accuracy of the milling device in positioning and orienting a tool.

- **Vibrations in the stock-in-progress**
  In practice sometimes vibrations of the stock-in-progress, due to cutting forces, result in a less accurate model.

- **Possible breakage of the desired shape**
  Due to the cutting forces very thin parts of the shape will probably break.

In Chapter 2 we will present our approach to the problem of sculpturing shapes by 6-axis milling. We propose a decomposition of the problem into the problem of extracting a limited number of tool access orientations from the free-form object geometry, and the problem of generating the milling paths given a fixed tool access orientation. The former problem is addressed in Chapter 3 and an algorithm to extract tool access orientations is described. The latter problem is addressed in Chapter 4 and a robust algorithm to generate tool paths is presented. In Chapter 5 a software implementation of our method for the Sculpturing Robot system is described. Chapter 6 provides an application and feasibility evaluation. Finally, in Chapter 7 conclusions are drawn with respect to the research issues and the requirements.

2

Shape sculpturing using 6-axis milling

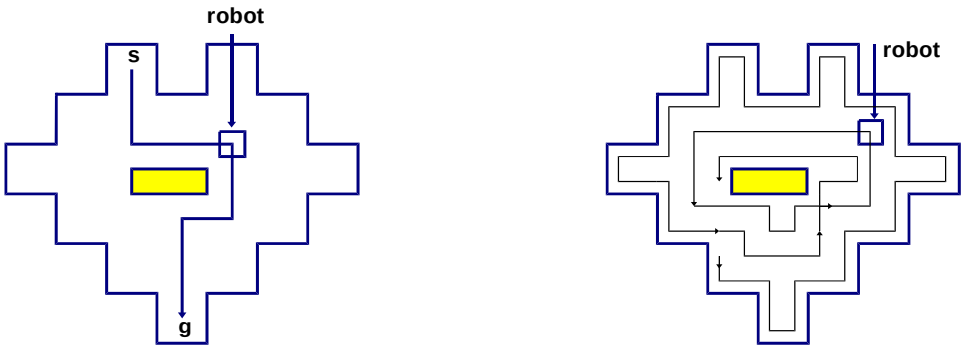
In this chapter we focus on the problem of sculpturing shapes from a stock of material using 6-axis milling. In Section 2.1 we describe shape sculpturing as a robot motion planning problem. In Section 2.2 we discuss the capabilities of the sculpturing process with regard to the shapes that can be machined exactly, as well as with regard to the complexity of the sculpturing process. Finally in Section 2.3, we propose a decomposition of the sculpturing problem into the problem of extracting a limited number of tool access orientations from the geometry of a free-form object, and the problem of generating the milling paths given a fixed tool access orientation.

2.1 Sculpturing as a robot motion planning problem

In this section, sculpturing is described as a robot motion planning problem in a geometric fashion. Before describing the sculpturing problem, we discuss two simpler problems: planning a path to move from a given start point to a goal point and planning a path to mow a lawn. For a thorough introdution on robot motion planning, we refer the reader to the book by (Latombe 1991), and the book by (de Berg, van Kreveld, Overmars & Schwarzkopf 1997), that describes the application of computational geometry algorithms to several fields, including robot motion planning.

To be able to perform a task by a robot, a motion plan for that robot has to be computed. To be able to plan a motion, some knowledge about the environment in which the robot is moving, has to be taken into account. For example, a mobile robot moving around in a factory must know where obstacles are located. In the field of robot motion planning most of the research deals with the *basic motion planning problem*: planning a path for a robot from a given start position to a given goal position without colliding with any of the obstacles. A more difficult problem is *the lawn-mower problem* in which a path is planned for a robot to mow a lawn completely. A path has to be planned such that the volume of the robot sweeps out the complete lawn. Figure 2.1 illustrates both motion planning problems.

In general there exists a multitude of paths with which the same task can be performed.



(a) A path from a given start point s to a given goal point g.

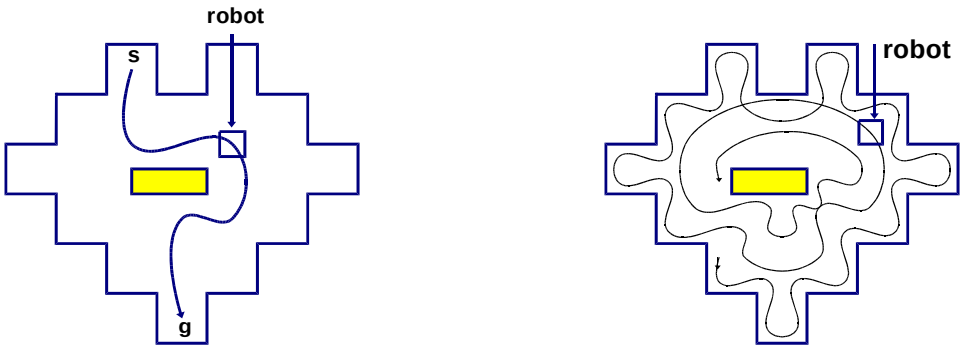
(b) A path for cutting a lawn.

Figure 2.1 Problems in robot motion planning. It is assumed that the robot consists of a square. Paths along which to move the centre of the square are shown.

The time to follow the path is a good criterion to compare paths. Often, the length of the path is not a good criterion. For example, some robots can only move in a straight line; they have to slow down, stop, and rotate, before they can start moving into a different directions, so any turn along the path causes some delay. For these robots not only the path length, but also the number of turns on the path has to be taken into account. Often a path is computed in two phases. In the first planning phase a collision-free path is computed and in the second smoothing phase the collision-free path is replaced by another collision-free path that can be followed by the robot much faster. But smoothing lawn-mower paths using a naive method does not work: Figure 2.2 shows that the robot, when following the smoothed path, will not completely mow the lawn. Since smoothing cannot be applied straightforward, we consider in this thesis only the generation of paths that consist of straight line segments.

An extension of the 2D lawn-mower problem is the 3D *sculpturing problem*. In the sculpturing problem a 3D free-form shape F should be carved, starting with a stock of material S by a milling tool T attached to a tool holder H. This task involves six degrees of freedom: three to position a tool reference point relative to the stock-in-progress and three to orient the tool and the tool holder. Hence, any point on a tool path should be specified by six values; three denoting a location and three denoting an orientation. In this chapter we describe the sculpturing process as general as possible without taking into account practical limitations, yet.

For every position on a path for which the tool intersects the stock, the stock is shrunken: the intersection of the tool with the stock is removed from the stock. To emphasis this dynamic change of the stock we often use the phrase "stock-in-progress" to denote the stock.



(a) A smoothed path from s to g.

(b) A smoothed path for mowing a lawn. Due to the smoothing not all grass is removed.

Figure 2.2 Smoothed paths.

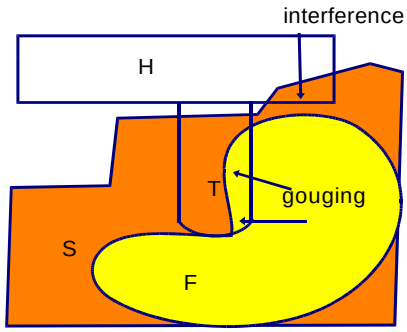


Figure 2.3 In the sculpturing problem both gouging between the tool and the shape to be sculptured, and interference between the tool holder and the stock-in-progress must be avoided.

This sculpturing process has to satisfy the following conditions (see also Figure 2.3):

1. The milling tool T should never gouge the shape F.
2. The tool holder H should never interfere with the stock-in-progress S.

The tasks in the lawn-mower problem and the sculpturing problem are similar: the lawn-mower has to sweep out the lawn and the milling tool T has to sweep out $S \backslash F$, *i.e.* the set difference of S and F. The important difference between the lawn-mower problem

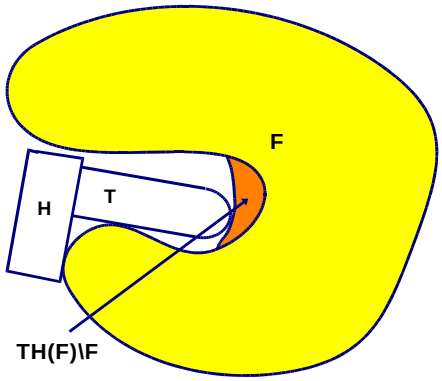


Figure 2.4 2D example of the TH-hull of a free-form shape F.

and the sculpturing problem is the dynamic effect of the volume removal process: the stock-in-progress is a shrinking obstacle for the tool holder. Hence, the existence of collision at a point on the path depends heavily on the path followed up to that point. Given a tool holder H, a milling tool T and a free-form shape F we describe in the next section a minimal shape enclosing F that can be obtained without violating condition 1 and 2.

2.2 On the capabilities of the sculpturing process

Due to the conditions 1 and 2 described in the previous section, it is not possible to obtain every free-form shape exactly. Often, depending on the shape of T and H, only a so-called TH-hull, containing the free-from shape F, can be sculptured (Figure 2.4). In this section we will define this hull as the result of a sculpturing process in which as much material as possible is removed from a stock S without violating condition 1 and 2. A similar concept has been introduced by (Edelsbrunner & Mücke 1994): they define the α-hull of a set of points F as the material that cannot be removed by a spherical eraser with radius α without intersecting F, *i.e.* without violating condition 1.

In practice, it may be impossible to obtain the TH-hull exactly, but instead a shape that contains the TH-hull is obtained; in other words, some material is not removed due to constraints different from condition 1 and 2. Additional constraints may be due to the particular mechanism used to perform the sculpturing process. For example, if the stock of material S is mounted on a turn table, the stock should include a support structure below S. No material from this support structure may be removed. This may inhibit the removal of other significant parts of S. In this chapter these constraints are not taken into account.

The TH-hull concept is a generalization of the α-hull concept. If H is the empty set and T is a sphere of radius α, then the TH-hull of F is equal to the α-hull of F.

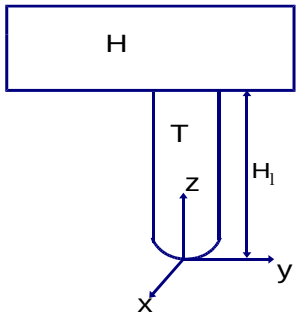


Figure 2.5 The tool T and the tool holder H are defineded in the same Cartesian coordinate system.

(Edelsbrunner & Mücke 1994) describe the α-hull of a set of points F as follows: Think of $\mathbb{R}^3$ filled with foam and the points of F made of more solid material, such as rock. Now imagine a free-flying spherical eraser with radius α. It is omnipresent in the sense that it carves out foam at all positions where it does not intersect with any of the sprinkled rocks, that is, points of F. The resulting object is called the α-hull of F.

We want to generalize this concept to describe the best result of machining with a free-flying milling device consisting of a tool holder H and a milling tool T attached to it.

Assume the shape is carved by removing material from a stock of foam. Since the tool holder may not interfere with the stock-in-progress, the description of this carving process is more complex then the carving process to obtain the α-hull. Since the stock-in-progress shrinks, the set of positions at which the tool can be positioned may grow. Hence, we cannot describe the carving process by only considering tool positions for which the tool holder does not interfere with the stock-in-progress. Instead, we assume that the tool may follow any finite tool path.

To represent the tool and tool holder volume we attach a Cartesian coordinate system to the tool as illustrated by Figure 2.5. The origin of this Cartesian coordinate system coincides with a point on the tool boundary volume. This point is called the tool reference point. The z-axis of this coordinate system denotes the tool direction. Further, the x-axis and y-axis are perpendicular to the z-axis. It is assumed that the tool holder is above the xy-plane, *i.e.* for all $(x, y, z) \in H$ holds $z > 0$.

Before we define the TH-hull, we show first that the convex hull of a shape F, defined as the intersection of all half-spaces enclosing F, can always be sculptured exactly.

Let $H_l = MIN\{z \mid (x, y, z) \in H\}$. Let a free-form shape F be given. Assume that an initial stock of material S contains the convex hull of F. Since $H_l > 0$, it is always possible to remove the material outside a half-space enclosing F layer by layer with layer thickness H_l by a process in which the holder is always outside the stock-in-progress and the z-direction of the tool is perpendicular to the half-space. Proceeding in this way for

all half planes that define the convex hull of F, we obtain this convex hull.

So, for the definition of the TH-hull we can assume that S equals the convex hull of F. Now we define the TH-hull as follows:

Definition 2.1 (TH-hull) *Given a shape F, a tool T and a tool holder H, the TH-hull of F is defined as follows:*

Let the initial stock of material be the convex hull of F. Let the tool be omnipresent in the sense that it can follow every finite interference-free tool path. We define the TH-hull of F as the intersection of all shapes that can be sculptured with one of these finite interference-free tool paths.

Since, in general the number of interference-free tool paths will be infinite this definition does not provide a practical method to obtain the TH-hull. It is an open question whether the TH-hull can be obtained by a finite interference-free tool path. Even if the TH-hull of a shape can be obtained by a finite tool path, its computation may take a long running time due to the relative large number of degrees of freedom involved (six). In general, see e.g., (Latombe 1991), the time complexity of path planning increases exponentially with the number of the degrees of freedom. Therefore, performing a direct search for collision-free milling paths in a 6-dimensional space is computationally expensive. Also, the sculpturing process should contain paths to insert the tool and the tool holder into a complex cavity. The problem of computing such paths has been widely studied in robotics, where it is referred to as the *peg-in-hole problem*. For example, (Joskowicz & Taylor 1996) describe research on efficiently computing an interference-free insertion path of a body into a cavity. In the worst case, the peg (the tool and the tool holder) and the hole (the cavity) are tightly fit, so that the clearance between the peg and the hole is small. Therefore, the computation of a single insertion path is already computationally expensive. Hence, a direct search in a 6-dimensional space as well as the peg-in-hole problem should be avoided in a computationally inexpensive sculpturing algorithm.

2.3 On the selected approach to the sculpturing problem

Sculpturing is a task that involves six degrees of freedom: it is allowed that the location as well as the orientation of the tool changes at any time in the milling process. To reduce the complexity of the sculpturing process the computation of the tool orientations and the tool locations is separated. First, from the geometry of the shape to be sculptured a limited number of tool orientations is extracted. Next, a number of milling stages is performed to obtain the shape. At each stage the principle of 3-axis milling is applied: one of the computed tool orientations is used as a fixed tool orientation and milling paths are generated to remove as much material as possible.

Given a tool orientation, we can use visibility and accessibility analysis to compute the region of a shape that can be accessed by the milling tool with that orientation. With this approach we neglect the peg-in-hole problem, because only visible regions of the shape

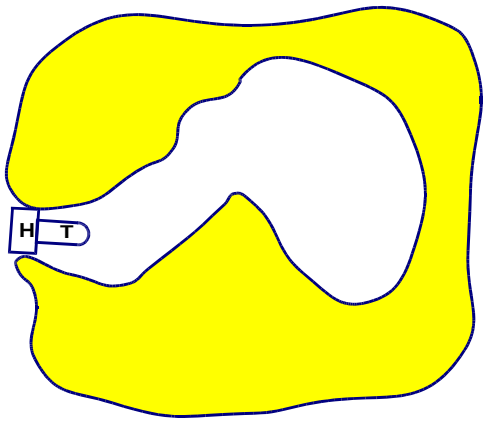


Figure 2.6 To access the cavity with the tool T and the tool holder H an insertion path has to be planned.

---

will be machined. Figure 2.6 shows a cavity that cannot be machined by our approach, because the planning of insertion paths would be necessary in that case. So we loose some of the power of the sculpturing system, but this is the price to pay for efficient computation.

Since fixing the tool orientation reduces the complexity of sculpturing, we decompose the sculpturing problem into an *accessibility problem* and a *path planning problem*:

Problem 2.1 (accessibility of a free-form shape) *Given a free-form shape F, a tool T, and a tool holder H, extract from F's geometry a limited number of tool orientations $\mathbf{o}_i$ to access F's boundary.*

Problem 2.2 (path planning) *Given a free-form shape F, a tool T, a tool holder H, the stock-in-progress S and a tool access orientation $\mathbf{o}_i$, generate tool paths to remove as much material as possible with orientation $\mathbf{o}_i$ satisfying the conditions:*

1. *The milling tool T should never gouge the shape F.*
2. *The tool holder H should never interfere with the stock-in-progress S.*

Hence, the complexity of the problem of sculpturing with six degrees of freedom is reduced such that computational feasibility is reached. By solving the two problems independently, a fully automated method for sculpturing shape models of free-form objects is obtained. These two problems will be addressed in Chapters 3 and 4, respectively.

3

Computation of tool access orientations by approximate accessibility analysis

In the previous chapter an approach to sculpturing based on a limited number of tool access orientations has been indicated. In this chapter we present a method to compute such a limited set of tool access orientations for the sculpturing task. It is assumed that the shape F to be sculptured is given by an initial free-form object consisting of a set of B-spline surfaces enclosing a volume. We allow the B-spline surface data to be as inaccurate as the user-specified geometric accuracy of the initial free-form object. Globally the approach we used works as follows. We transform the B-spline surface representation into a voxel representation. This voxel representation consists of a 3D array of unit cubes (called voxels). For each B-spline surface a number of sample points are generated. Each voxel that contains a sample point is identified as a boundary voxel. For each of these boundary voxels a visibility map is computed. This visibility map of a voxel v contains a set of directions from which the part of F's boundary enclosed by v is visible. For each voxel v its visibility map is transformed to an accessibility map that contains a set of directions to access the part of F's boundary enclosed by v. From these accessibility maps a limited number of tool access directions are extracted to sculpture the shape. Finally, each tool access direction is extended to a tool access orientation. The details of the approach will be worked out in the following sections.

3.1 Previous work

(Gan 1992) applies visibility analysis to determine a range of set-up directions, that allows a given surface to be manufactured by a 3-axis NC machine without having to redirect the stock-in-progress. Such a range of directions can be represented by the (global) visibility map, abbreviated to V-map, of the surface. Any point in a V-map represents a direction such that the entire surface is visible from infinity. Each direction can be represented by a vector of length 1. Therefore, a V-map is a subset of the unit sphere, *i.e.* the sphere of radius 1. A two-stage approach is proposed for finding the V-map of a surface. An initial

range of directions, termed the local V-map, is first obtained by considering only the local geometry of the shape boundary. The (global) V-map is then obtained by removing from the local V-map every direction that interferes with other parts of the shape. (Spyridi & Requicha 1990) use local and global visibility cones and a similar two-stage approach to determine accessibility of surface features for coordinate measuring machines. (Woo 1994) applies the concept of V-maps to a wide range of manufacturing processes. (Chen, Chou & Woo 1993, Gupta et al. 1996, Tang, Woo & Gan 1992) present geometric algorithms that can be used to select a direction maximizing the number of surfaces that are visible with that direction. (Elber & Cohen 1995) present a method to compute the local V-map of a surface exactly. (Elber 1994) considers the accessibility problem for 5-axis milling of a convex surface S positioning a flat-end perpendicular to S. He presents an approach that reduces this 5-axis milling accessibility problem to a 3-axis milling accessibility problem taking into consideration other surfaces that limit the accessibility of F.

In all references above, the V-maps are stored per surface. Hence, accessibility for a given tool access direction can be determined only per surface or for a surface set and not for regions, whose boundaries in general do not coincide with surface boundaries. Furthermore, to access surfaces like spheres and cylinders, more than one tool direction is needed. So, for such a surface its visibility map is empty. Also, the V-maps are computed exactly as subsets of the unit sphere. Therefore, a computationally expensive procedure is required to compute them.

Our approach embeds the shape in a voxel data structure and computes a V-map for each voxel that intersects the shape boundary. Since this approach requires the computation of many V-maps, an efficient procedure to compute V-maps is needed. Therefore, we replace the exact computation of V-maps by an approximate computation. We do not select tool access directions from the unit sphere, but from a finite set of directions that are distributed almost uniformly on the unit sphere.

To the best of our knowledge there is only one reference in the literature on tool accessibility computations using an approximate approach: (Kang & Suh 1997) decompose the unit sphere as well as each surface into a finite number of triangular patches. For each surface they compute a binary spherical map BSM, such that the Boolean value $BSM(i, j)$ denotes whether the ith surface patch is accessible from the jth spherical patch.

3.2 Mathematical background

3.2.1 On the shape model obtained by the milling process

A ball-end milling tool T can be modelled as a cylinder holding a spherical eraser, both with radius α. In this subsection we temporarily neglect interference between the tool cylinder and the shape F, *i.e.* we assume that, wherever the spherical eraser does not intersect F, also a global tool access direction exists such that the cylinder holding the spherical eraser does not intersect F. Since interference between the spherical eraser and the shape has to be avoided, in general it will be impossible to physical reproduce the shape F exactly. Recall from Chapter 2, that it is only possible to carve out the α-hull

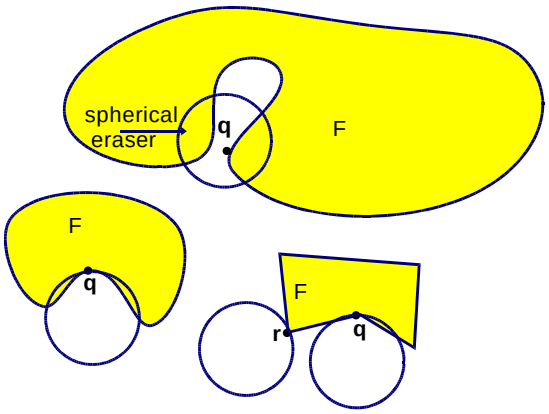


Figure 3.1 An example of a violation of accessibility condition 1 (top) and two examples (bottom) of a violation of accessibility condition 2. In the left bottom example the curvature exceeds $1/\alpha$ at $\mathbf{q}$, where ∂F is locally concave. In the right bottom example the curvature of the concave surface intersection at $\mathbf{q}$ is ∞ (violation of condition 2), and the curvature of the convex surface intersection at $\mathbf{r}$ it is $-\infty$ (no violation).

$\mathcal{H}_\alpha(F)$ of F. $F = \mathcal{H}_\alpha(F)$ if the spherical eraser can touch every point on F's boundary ∂F without intersecting F, *i.e.* if the following two accessibility conditions hold.

1. If the spherical eraser intersects the interior of F, then the intersection of ∂F and the spherical eraser is homeomorphic to an open disk in $\mathbb{R}^2$.

2. The curvature at each point of ∂F is at most $1/\alpha$, where we define the curvature of a point of ∂F negative if the boundary at that point is locally convex and positive if the boundary is locally concave.

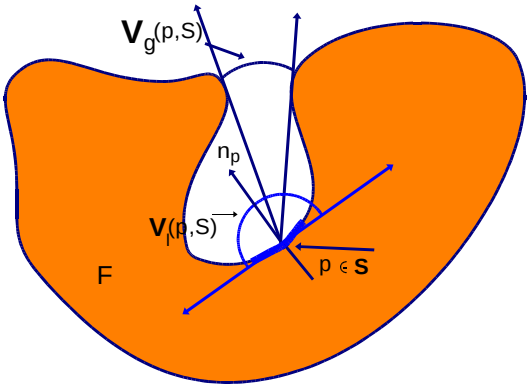
Figure 3.1 illustrates cases for which $F \neq \mathcal{H}_\alpha(F)$. Condition 1 is violated if F contains small holes. Condition 2 in the case of a strongly curved surface and in the case of two intersecting surfaces that define a concave part of the boundary. Later in this chapter we show that our data structure cannot distinguish convex and concave surfaces intersection. Therefore, we must assume that the boundary in the neighborhood of a surface intersection is never accessible.

3.2.2 Definitions

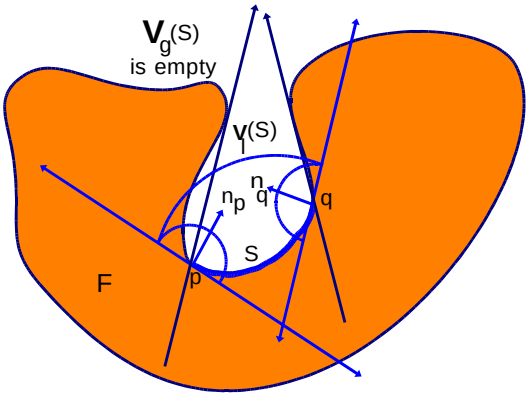
From Chapter 2 we recall the accessibility problem: Given a complex three-dimensional shape $F \subset \mathbb{R}^3$ to be machined, provide a limited set of tool access directions to access F's boundary.

We present a number of definitions to formalize the accessibility concept.

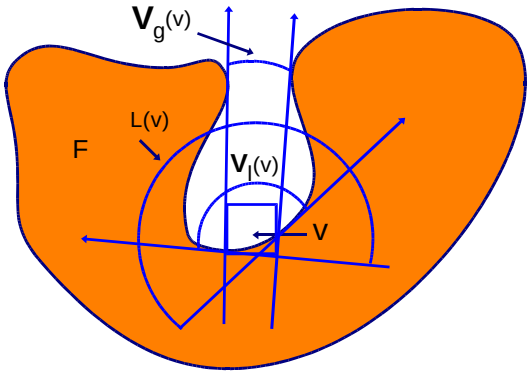
Let $F \subset \mathbb{R}^3$ be a three-dimensional shape defined by a surface set $\mathcal{F}$. For our presentation it is convenient to assume that F emits rays. Further, we consider only those normal vectors on ∂F that point outwards F.



(a) V-maps for a point. $V_l(p, S)$ denotes the hemisphere $H(n_\mathbf{p})$, where $n_\mathbf{p}$ denotes the normal at point $\mathbf{p}$ on the surface S.



(b) V-maps for a surface. $V_l(S)$ consists of the intersection of all hemispheres $H(n_\mathbf{p})$, for each $\mathbf{p} \in S$. Note that the global V-map is empty.



(c) V-maps for a voxel. $V_l(v)$ consists of the intersection of all hemispheres $H(n_\mathbf{p})$, for each $S \in \mathcal{F}$, $\mathbf{p} \in v \cap S$. The light map $L(v)$ consists of the union of all hemispheres $H(n_\mathbf{p})$ for each $S \in \mathcal{F}$, $\mathbf{p} \in v \cap S$.

Figure 3.2 2D examples of V-maps.

Figure 3.2 illustrates the visibility definitions given below. A global visibility map will be defined for a single point $\mathbf{p}$ on a surface S, for the set of all points on a surface S and for all surface points contained in a voxel v.

For a surface point $\mathbf{p}$ on a surface S, the global visibility map $V_g(\mathbf{p}, S)$ represents the directions of rays that start at $\mathbf{p}$ and do not intersect F.

For a surface S, the global visibility map $V_g(S)$ represents the directions for which

every ray starting at a surface point on S does not intersect F, *i.e.*

$$V_g(S) = \bigcap_{\mathbf{p} \in S} V_g(\mathbf{p}, S).$$

For a boundary voxel v, $V_g(v)$ represents the directions for which every ray starting at a surface point contained in v does not intersect F, *i.e.*

$$V_g(v) = \bigcap_{S \in \mathcal{F}, \mathbf{p} \in v \cap S} V_g(\mathbf{p}, S).$$

To define local visibility maps we consider the local geometry of a surface rather than the whole shape F. The normal $\mathbf{n_p}$ at a point $\mathbf{p}$ on a surface gives one direction in which the point is visible from infinity if the ray has no intersection with F. However, the same point may be visible from many other directions, up to a hemisphere $H(\mathbf{n_p})$ of directions bounded by the tangent plane at $\mathbf{p}$. Therefore, we define the local visibility map $V_l(\mathbf{p}, S)$ of a point $\mathbf{p}$ on a surface S with normal $\mathbf{n_p}$ by $V_l(\mathbf{p}, S) = H(\mathbf{n_p})$. Note that for an intersection point of multiple surfaces different local visibility maps may exist.

Analogous to the global visibility maps we define the local visibility maps of surfaces and boundary voxels as

$$V_l(S) = \bigcap_{\mathbf{p} \in S} V_l(\mathbf{p}, S)$$

and

$$V_l(v) = \bigcap_{S \in \mathcal{F}, \mathbf{p} \in v \cap S} V_l(\mathbf{p}, S).$$

For a boundary voxel v let the light map $L(v)$ represent all directions of rays starting at any surface point contained in v, *i.e.*

$$L(v) = \bigcup_{S \in \mathcal{F}, \mathbf{p} \in v \cap S} V_l(\mathbf{p}, S).$$

We define accessibility of a point on ∂F as follows:

Definition 3.1 *A shape F is accessible at a point $\mathbf{q}$ on ∂F by a ball-end milling tool T from direction $\mathbf{d}$ if and only if*

- *the spherical eraser can touch $\mathbf{q}$, i.e. $\mathbf{q} \in \partial \mathcal{H}_\alpha(F)$, and*
- *the tool cylinder does not intersect the shape F.*

Figure 3.3 illustrates this definition. It can be generalized to voxels as follows:

Definition 3.2 *A voxel v containing a part of the boundary of the shape ∂F is accessible by a ball-end milling tool T from direction $\mathbf{d}$ if F is accessible at every point on $\partial F \cap v$ by T from direction $\mathbf{d}$.*

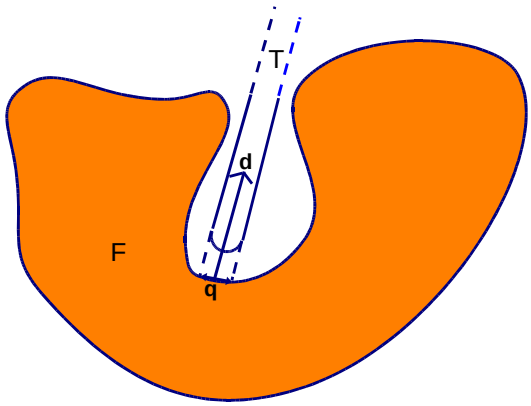


Figure 3.3 F is accessible at $\mathbf{q}$ by T from direction $\mathbf{d}$ (2D example).

3.3 Validity of the initial B-spline model

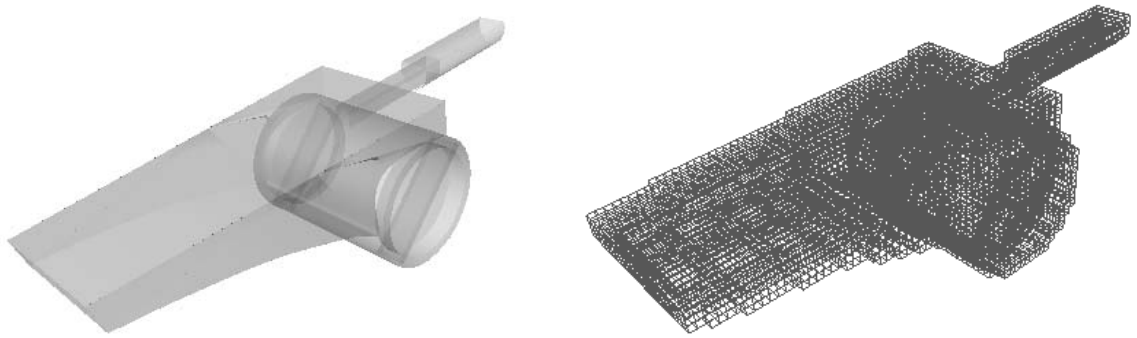
In the conceptual design phase the initial shape geometry may be only roughly specified. We consider the approach in which, with help of a B-spline modeler, a free-form shape F is defined as the volume enclosed by a set of B-spline surfaces. For this initial model we allow the B-spline surface data to be as inaccurate as a user-selected geometric accuracy. The surfaces may be self-intersecting or intersect each other, and parts of the surfaces may be no part of the boundary of F. If a surface is part of the boundary, it need not be specified which side of the surface is the outside of the object. Also surfaces that are intended to be adjacent, are allowed to be separated by narrow gaps as long as these gaps are smaller than the user-selected geometric accuracy.

Our notion of a B-spline surface set is less restrictive than the notion of a boundary model. A boundary model represents a solid as a union of faces, bounded by edges, which in turn are bounded by vertices. In the general boundary model the shape of a face is represented by a free-form surface and the shape of an edge by a 3D curve. For an extensive description of a boundary model we refer the reader to the textbook by (Mortenson 1997). Since a boundary representation does not support the representation of roughly specified shape geometries we selected a representation that extends the B-spline surface set with a voxel representation.

We assume that the initial object represents a shape F by a B-spline surface set that approximately encloses a volume within a user-specified spatial accuracy d. First, we transform the B-spline surface set into a voxel representation. We use the colors white, grey and black to distinguish exterior, boundary and interior voxels. All voxels that intersect one of the B-spline surfaces are colored grey. The voxels that are enclosed by the grey voxels are colored black. The remaining voxels are colored white.

We state the following requirements on the validity of an initial free-form object.

1. The grey voxels represent one boundary, *i.e.* they are connected.



(a) Initial model with overlapping B-spline surfaces.

(b) Voxel representation with a low resolution.

Figure 3.4 Initial CAD model and voxel representation for a dust buster.

2. The grey voxels enclose at least one volume, *i.e.* there are black voxels. Since we allow part of the B-spline surfaces to be in the interior of an initial free-form object, it is possible that these surfaces divide the interior in more than one volume.

3. Exactly one side of a surface should define the outside of the free-form object, *i.e.* only at one side of a surface the normal vectors should point outwards.

The voxel representation and the procedure to check these conditions are described in the remainder of this section.

A 3D array of voxels is generated that encloses the shape F. Surface sampling is used to identify the boundary voxels that contain parts of the boundary ∂F of F. The voxels that are not boundary voxels are either exterior voxels or interior voxels. F encloses the interior voxels. Figure 3.4 shows an example of a voxel model of an initial CAD model. The user-selected geometric accuracy of the physical shape model is taken as the voxel resolution d (*i.e.* the voxel edge length).

We use a triple (i, j, k), where $i, j, k \in \mathbb{Z}$, to identify a voxel $[(i - 1/2)d, (i + 1/2)d] \times [(j - 1/2)d, (j + 1/2)d] \times [(k - 1/2)d, (k + 1/2)d]$.

A distinction is made between the different kinds of neighbor of a voxel. Two voxels can share either a common vertex, a common edge, or a common face as shown in Figure 3.5. A line can be approximated by voxels, such that neighbors share at least one vertex. A surface can be approximated by voxels, such that neighbors share at least one edge. A volume can be approximated by voxels, such that neighbors share at least one face. Thus, two voxels (i_1, j_1, k_1) and (i_2, j_2, k_2) are called **vertex-neighbors** if and only if the absolute difference in three corresponding coordinates is 1. Two voxels are called **edge-neighbors** if and only if the absolute difference in two of the corresponding coordinates is 1 while the difference in the other coordinate is 0. Two voxels are called **face-neighbors**

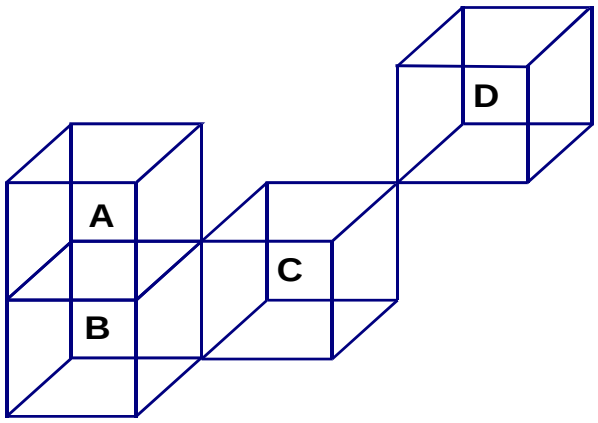


Figure 3.5 A and B are face neighbors, B and C are edge neighbors and C and D are vertex neighbors.

---

if and only if the absolute difference in one of the corresponding coordinates is 1 while the difference in the other two coordinates is 0. Two voxels are **vertex-adjacent** if they are vertex-, edge- or face-neighbors. Two voxels are **edge-adjacent** if they are edge- or face-neighbors. Two voxels are **face-adjacent** if the are face-neighbors. A **face-**, **edge-**, **vertex-path** from voxel v_0 to voxel v_m is a sequence of voxels $(v_0, v_1, \ldots, v_{m-1}, v_m)$ where v_i is adjacent to v_{i+1}, $i = 1, \ldots, m-1$ according to the corresponding adjacency. Two voxels are **connected** if and only if they have the same color and there is at least one path from one to the other, where every element of the path has the same color as the given voxels. A maximal set of face-connected voxels is a **face-component**. A maximal set of edge-connected voxels is a **edge-component**. Note that an edge component consist of one or more face-components, each connected by an edge.

By definition a voxel representation represents a valid volume if its boundary is represented by one grey edge-connected component GEC and its volume is represented by one black face-connected component BFC. We compute these components with help of the paint-spreading Algorithm 3.1 from (Dijkstra 1959). This algorithm determines, given a voxel v, a color c and an adjacency a, the a-component containing v, and paints all voxels of the component with the color c.

We assume that the voxels and their colors are given by a 3D array $V(i,j,k)$, where $i = i_{min} \ldots i_{max}$, $j = j_{min} \ldots j_{max}$, and $k = k_{min} \ldots k_{max}$. $V(i,j,k)$ has as color $WHITE$, $GRAY$ or $BLACK$. $\{(i,j,k) \mid i_{min} \leq i \leq i_{max} \wedge j_{min} \leq j \leq j_{max}, k_{min} \leq k \leq k_{max})\}$ is called **the raster**. The **raster border** is the set
$R = \{(i,j,k) \in \mathrm{R} |\ i = i_{min} \vee i = i_{max} \vee j = j_{min} \vee j = j_{max} \vee k = k_{min} \vee k = k_{max}\}$.

We require that voxels on the raster boundary do not contain surface points. The following procedure obtains a volume representation from a B-spline surface set.

1. **Determine the raster sizes**
   Since a B-spline surface is contained in the convex hull of its control points, the

Algorithm 3.1 Dijkstra's paint-spreading algorithm

Given a voxel v, an adjacency a and a color c, Dijkstra's paint-spreading algorithm determines the a-component that contains v and paints all voxels in that component with color c. This algorithm starts with painting v with the color c. Next the color is spread out over the component. If a voxel is painted it is added to a list of voxels L. If the voxel is removed from L, its paint is spread to its neighbors. Hence, L denotes the border between the painted and unpainted part of the component. Because each voxel is colored at most once, the running time of Dijkstra's algorithm is linear in the number of voxels that change color.

```
Initialize L := (v) and paint v with color c;
while L ≠ ∅ do
  Select an element e from L;
  Remove e from L;
  Add all a-neighbors of e to L and paint these neighbors with color c;
end while
```

raster sizes can be derived from the minimal and maximal x, y and z values of the control points, such that the raster boundary does not contain surface points.

2. **Initialize the raster as black**

3. **Sample the B-spline surfaces with a resolution** d
   Paint each voxel that contains a sample point grey.

4. **Paint the outside voxels white**
   Select a voxel v on the raster boundary. Paint with Algorithm 3.1 the face-component containing v white.

5. **Determine the number of GEC's and BFC's**
   Now, the volume of the B-spline model is represented by all black voxels and the boundary of the volume is represented by all grey voxels. Next in Algorithm 3.2 we apply Dijkstra's paint spreading algorithm to count the number of grey edge components (GEC's) and black face components (BFC's). Because each voxel is colored at most once, the running time of the algorithm is linear with the number of voxels in the raster.

Requirement 1 on the validity of the initial object is satisfied if we find exactly one GEC and requirement 2 is satisfied if we find at least one BFC. Requirement 3 is tested as follows. The normal vectors on a B-spline surface $S(u,v)$ are given by $\mathbf{n}(u,v) = SIGN(u,v)\frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$, where either $SIGN(u,v) = -1$ or $SIGN(u,v) = 1$, such that $\mathbf{n}(u,v)$ points outwards. We use surface sampling to compute $\frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$ at a number of sample points. We implemented a robust method to obtain a sufficiently dense set of surface points. For details about this method we refer the reader to (Vergeest, Broek, Schierbeek & Tangelder 1991). For $SIGN(u,v) = 1$ and for $SIGN(u,v) = -1$ we

Algorithm 3.2 Algorithm for counting the number of GEC's and BFC's

Let $GECCounter$ denote the number of GEC's found.
Let $BFCCounter$ denote the number of BFC's found.
Let $VoxRep$ denote a copy of the voxel representation.

```
GECCounter := 0;
while VoxRep contains grey voxels do
  Select a grey voxel v from VoxRep;
  Paint with Dijkstra's algorithm the face-component containing v white;
  GECCounter := GECCounter + 1;
end while
BFCCounter := 0;
while VoxRep contains black voxels do
  Select a black voxel v from VoxRep;
  Paint with Dijkstra's algorithm the face-component containing v white;
  BFCCounter := BFCCounter + 1;
end while
```

compute the number of normal vectors pointing towards a white voxel. If both numbers are more than zero, requirement 3 is violated, because at both sides of the surface normal vectors that point outwards are found.

In the next section our method to determine accessibility is described.

3.4 Method to determine accessibility

In this section we assume that a B-spline surface set $\mathcal{F}$ of an initial model of a shape F is given. Furthermore, we assume that a valid voxel representation with a user defined resolution d of F has been obtained with the method that has been described in the previous section. In this section we present a method to compute for each voxel its global voxel accessibility map, and we describe an algorithm to extract a limited number of tool access directions from these maps to sculpture the shape F. The method consists of the steps below. Step 1, 3 and 4 will be described in a subsection in more detail. To obtain precise results a much smaller sample distance is used than for the validation method of the previous section. Therefore, it is also necessary to compute the boundary, interior and exterior voxels again.

1. Apply surface sampling with a sample distance $d' \ll d$ to determine the boundary voxels and to compute at each boundary voxel v, an approximate local V-map $V_l'(v)$ and an approximate light map $L'(v)$. Let F' denote the obtained voxel representation of the shape F. Use a tessellation of the unit sphere as described in Section 3.4.1 to record these maps. Also record at each boundary voxel v the maximal curvature and maintain an index to the surface from which v contains sample points. Set this index to *null* if v contains sample points from more than one surface. Recall

from Section 3.2.1 that a boundary voxel v is considered to be inaccessible if the curvature is too large or if it contains more than one surface.

2. Determine the interior, boundary and exterior voxels with Algorithm 3.1.

3. Compute for each voxel v, the approximate global visibility map $V_g'(v)$ using the approximate local visibility map $V_l'(v)$ and the approximate light maps $L'(w)$ for all boundary voxels $w \neq v$.

4. Transform for each voxel, its approximate global visibility maps to an approximate map containing global tool access directions.

3.4.1 Applying surface sampling to store the local V-maps and light maps using a tessellation of the unit sphere

In (Gan 1992) it is proposed to derive the local V-map of a surface from the Gaussian map (G-map), which is a map of the surface normals on the unit sphere U. The local V-map can be constructed by the intersection of hemispheres, each having as its pole a point on the G-map. This V-map can be computed as a subset of U in time $O(n\,log\,n)$, where n is the number of normals on the spherical convex hull of the G-map (Gan 1992, Woo 1994). First a surface is sampled and its G-map is built, next its V-map is built from the G-map, finally its G-map can be deleted.

Because we build a V-map per voxel, it would be necessary to store a G-map per voxel. To avoid storing the G-maps we integrate surface sampling with building the V-maps. This is possible if we represent V-maps and light maps as subsets of a set N containing a finite number of unit vectors from U following the scheme presented below. Let (V, T) denote a tessellation of U, where $V \subset U$ is a set of vertices and T is a set of tessels. A tessel $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ is a triangular surface patch on U between the vertices $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. Let $N = \{\mathbf{n} \in U \mid \exists(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \in T : \mathbf{n} \perp \mathbf{v}_1 - \mathbf{v}_2 \wedge \mathbf{n} \perp \mathbf{v}_2 - \mathbf{v}_3\}$. Hence, N is a finite subset of U, that contains per tessel exactly one vector. This vector points outwards the unit sphere and is perpendicular to the triangle $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$.

When a sphere is tessellated, ideally we would like the tessels to be symmetrical, be identical in shape, and possess equal areas. Such a tessellation can only be obtained with help of one of the five Platonic polyhedra (Pugh 1976). The icosahedron (see Figure 3.6 left) is the Platonic polyhedron with the maximal number of identical faces. It contains 20 triangular faces. In order to obtain smaller cell resolutions with triangular tessels that are approximately equal in area and shape, q-frequency polyhedra can be generated using the "alternate method" (see e.g. (Pugh 1976)). The "alternate method" divides each face of the icosahedron in q^2 triangular faces by lines running parallel to the original edges of the triangles. Each triangle $\triangle \mathbf{v_1 v_2 v_3}$ of the q-frequency icosahedron is projected on the unit sphere to the tessel between $\mathbf{v_1}/|\mathbf{v_1}|$, $\mathbf{v_2}/|\mathbf{v_2}|$ and $\mathbf{v_3}/|\mathbf{v_3}|$, where $|\mathbf{v}|$ denotes the length of the vector v. Hence, we obtain a finite set $N \subset U$ containing $20q^2$ vectors which are almost uniformly distributed on the unit sphere. (Chen & Kak 1989) have described and applied this method in the development and implementation of a robot vision system

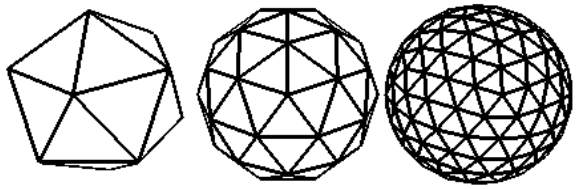


Figure 3.6 Three tessellations of the unit sphere obtained with an icosahedron (left), a two-frequency icosahedron (middle) and a four-frequency icosahedron (right).

for recognizing 3D objects. Figure 3.6 shows a one-, two- and four-frequency icosahedron tessellation of the unit sphere. Given a vector $\mathbf{n} \in U$, the tessel that contains $\mathbf{n}$ can be determined in $O(1)$ running time[1].

We approximate local V-maps and light maps of voxels v and hemispheres of tessels t by subsets $V_l'(v)$, $L'(v)$ and $H'(t)$ of N. For each vertex $\mathbf{v} \in V$ we compute the hemisphere with pole $\mathbf{v}$ as a subset of N by $H'(\mathbf{v}) = \{\mathbf{p} \in N \mid \mathbf{p} \cdot \mathbf{v} \geq 0\}$, where "$\cdot$" denotes the dot product.

For each tessel $t = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ we approximate its visibility hemisphere by $H_V'(t)$ defined as

$$H_V'(t) = H'(\mathbf{v}_1) \cap H'(\mathbf{v}_2) \cap H'(\mathbf{v}_3)$$

and its light hemisphere by $H_L'(t)$ defined as

$$H_L'(t) = H'(\mathbf{v}_1) \cup H'(\mathbf{v}_2) \cup H'(\mathbf{v}_3).$$

Note that the sets $H_V'(t)$ and $H_L'(t)$ can be preprocessed.

The local visibility map of a voxel v should only contain vectors for which the boundary of a voxel v is completely visible. Since these voxel visibility maps are obtained as the intersection of a number of approximate visibility hemispheres, we require that for a tessel t and a vector $\mathbf{n} \in t$, $H_V'(t)$ is a subset of its local visibility map $H'(\mathbf{n})$. This property is proved in the following lemma.

Lemma 3.1 *Let $t = (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$ denote a tessel. Let $\mathbf{n} \in t$. Then $H_V'(t) \subseteq H'(\mathbf{n})$.*

Proof: Let $\mathbf{w}_n = \lambda\mathbf{n}$ with $\lambda > 0$ denote the point at which $\triangle\mathbf{v_1}\mathbf{v_2}\mathbf{v_3}$ and $\lambda\mathbf{n}$ intersect. Since $\mathbf{w}_n$ is contained in the triangle with vertices $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ we can write
$\mathbf{w}_n = \alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3$, where $\alpha, \beta, \gamma \geq 0$, $\alpha + \beta + \gamma = 1$.
Hence, $\mathbf{n} = \alpha/\lambda\mathbf{v}_1 + \beta/\lambda\mathbf{v}_2 + \gamma/\lambda\mathbf{v}_3$. Therefore,

$$\begin{aligned} H'(\mathbf{n}) &= \{\mathbf{p} \in N \mid \mathbf{p} \cdot \mathbf{n} \geq 0\} \\ &= \{\mathbf{p} \in N \mid \alpha/\lambda\mathbf{p} \cdot \mathbf{v}_1 + \beta/\lambda\mathbf{p} \cdot \mathbf{v}_2 + \gamma/\lambda\mathbf{p} \cdot \mathbf{v}_3 \geq 0\}. \end{aligned}$$

[1] See (van den Belt & Tangelder 1997*a*).

From

$$\mathbf{p} \in H'_V(t) = H'(\mathbf{v}_1) \cap H'(\mathbf{v}_2) \cap H'(\mathbf{v}_3)$$

follows

$$\mathbf{p} \cdot \mathbf{v}_1 \geq 0 \wedge \mathbf{p} \cdot \mathbf{v}_2 \geq 0 \wedge \mathbf{p} \cdot \mathbf{v}_3 \geq 0.$$

Hence, $\alpha/\lambda \mathbf{p} \cdot \mathbf{v}_1 + \beta/\lambda \mathbf{p} \cdot \mathbf{v}_2 + \gamma/\lambda \mathbf{p} \cdot \mathbf{v}_3 \geq 0$ and $\mathbf{p} \in H'(\mathbf{n})$.
End of proof

The local light map of a voxel v should contain vectors for which part of the boundary of a voxel v is visible. Since these voxel light maps are obtained as the union of a number of approximate light hemispheres, we require that for a tessel t and the vector $\mathbf{n} \in t$, $H'_L(t)$ is a superset of its local visibility map $H'(\mathbf{n})$. This property is proved in the following lemma.

Lemma 3.2 *Let $t = (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$ denote a tessel. Let $\mathbf{n} \in t$. Then $H'_L(t) \supseteq H'(\mathbf{n})$.*

Proof: Let $\mathbf{p} \in H'(\mathbf{n})$. In the proof of lemma 3.1 we have shown that there exists $\alpha, \beta, \gamma \geq 0$ and $\lambda > 0$ such that

$$\mathbf{n} = \alpha/\lambda \mathbf{v}_1 + \beta/\lambda \mathbf{v}_2 + \gamma/\lambda \mathbf{v}_3.$$

From $\mathbf{p} \in H'(\mathbf{n})$ follows

$$\alpha/\lambda \mathbf{p} \cdot \mathbf{v}_1 + \beta/\lambda \mathbf{p} \cdot \mathbf{v}_2 + \gamma/\lambda \mathbf{p} \cdot \mathbf{v}_3 \geq 0.$$

Therefore,

$$\mathbf{p} \cdot \mathbf{v}_1 \geq 0 \vee \mathbf{p} \cdot \mathbf{v}_2 \geq 0 \vee \mathbf{p} \cdot \mathbf{v}_3 \geq 0$$

holds. Hence, $\mathbf{p} \in H'(\mathbf{v}_1) \cup H'(\mathbf{v}_2) \cup H'(\mathbf{v}_3) = H'_L(t)$.
End of proof

The approximate local V-maps $V'_l(v)$ and the approximate light maps $L'(v)$ are computed as follows:

- Initialize for each voxel v, $V'_l(v)$ with N and $L'(v)$ with $\emptyset$.
- Perform surface sampling with sample distance $d' \ll d$. For each sample point determine the voxel v containing it. Determine the normal $\mathbf{n}$ at the sample point as described at the end of this subsection and the tessel t whose patch contains $\mathbf{n}$. Intersect $V'_l(v)$ with $H_V(t)$ and unify $L'(v)$ with $H_L(t)$. If v contains sample points from only one surface then record an identifier of this surface. If v contains sample points from multiple surfaces then set the identifier to *null*. Also record at each voxel v the maximal curvature, which is computed as described at the end of this subsection.

Note that with this scheme subsets of N, like visibility maps, light maps and hemispheres, can be implemented by variables that require k bits storage, where $k = |N|$. For $N = \{\mathbf{n}_1, \mathbf{n}_2, \cdots \mathbf{n}_k\}$, the i^{th} bit of such a variable indicates whether the subset contains $\mathbf{n}_i$. Intersection and union can be implemented with the bitwise AND operation and the bitwise OR operation, respectively. Hence, this step has $O(kn)$ running time, where n is the number of sampled points.

Computation of normal vectors and curvature

The surfaces are sampled and the sample points are stored in a 2D array $\mathbf{p}(i, j)$. From these sample points we derive the partial u-, v-, uu-, uv- and vv-derivative at $\mathbf{p}(i, j)$. The normal vector is computed as the cross-product of the u- and v-derivative value at $\mathbf{p}(i, j)$. The partial u-, v-, uu-, uv- and vv-derivatives values at $\mathbf{p}(i, j)$ are used to compute the maximal surface curvature at $\mathbf{p}(i, j)$ with the method described by (Boehm 1996). Since for some situations partial derivatives computed analytically are vanishing, we prefer this geometrical approach above an analytical approach.

3.4.2 Obtain the global V-maps

The algorithm to obtain for each voxel v its global visibility map $V_g(v)$ is based upon the following observation.

Observation 3.1 *Let* $\mathbf{n} = (\mathbf{q} - \mathbf{p})/|\mathbf{q} - \mathbf{p}|$ *be a ray that leaves the voxel representation of the shape* F' *at a point* $\mathbf{p} \in v$ *and intersects* F' *again at a point* $\mathbf{q} \in w$*, where* v *and* w *denote boundary voxels. Then* $\mathbf{n} \in V_l(v)$ *and* $-\mathbf{n} \in L(w)$.

Given k directions and a voxel resolution d, Algorithm 3.3 obtains an approximate global V-map $V'_g(v)$ for all boundary voxels v. In this algorithm a beam of rays starting at v with directions $\mathbf{n}$ is used to denote the set of half lines starting in the voxel v and leaving v in the direction of $\mathbf{n}$. The running time is $O(km_1m_2)$, where m_1 is the number of boundary voxels of F' and m_2 is the average number of voxels intersected by a beam of rays. Figure 3.7 illustrates this algorithm.

3.4.3 Obtain a voxel map containing tool access directions

Let the milling tool T be positioned with access direction $\mathbf{n}$. Let $\partial T_{\mathbf{n}}$ denote the boundary of the spherical eraser outside the tool cylinder. Hence, $\partial T_{\mathbf{n}}$ is a hemisphere with pole $-\alpha\mathbf{n}$. If T is positioned anywhere such that $\partial T_{\mathbf{n}}$ intersects a voxel v, it may intersect also other voxels. Let $E_{\mathbf{n}}(v)$ be the set consisting of the voxel v and all the other voxels that $\partial T_{\mathbf{n}}$ may intersect. If $d \geq 2\alpha$, where d denotes the user defined voxel resolution and α the tool radius, then $E_{\mathbf{n}}(v)$ consists of the voxels that share at least a vertex with v. Selecting an appropriate value for d offers a trade-off between the cubic growth in storage against the accuracy of the tool access computations.

Algorithm 3.4 computes for each boundary voxel v an accessibility map $A'(v)$ containing suitable tool access directions. This algorithm has $O(km_1)$ running time, where m_1 is

Algorithm 3.3 Algorithm to compute global accessibility of the boundary voxels

```
for all boundary voxels v do
  V'_g(v):=V'_l(v);
  for all directions n ∈ V'_g(v) do
    compute the beam b of rays starting at v with direction n;
    for all voxel w intersected by b do
      if w is an interior voxel or w is a boundary voxel with −n ∈ L'(w) then
        V'_g(v) := V'_g(v)/{n};
      end if
    end for
  end for
end for
```

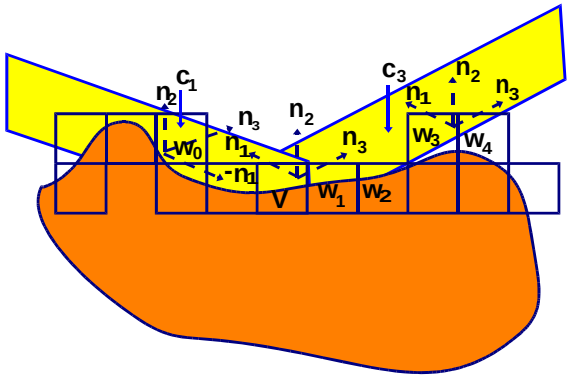


Figure 3.7 Obtaining the global V-maps. The local visibility map of v contains normal vectors n_1, n_2 and n_3. The light map of w_0 contains $-n_1$. The beam of rays c_1 with direction n_1 intersects w_0. Hence, n_1 is deleted from the global visibility map of v. The light maps of w_1, w_2, w_3 and w_4 do not contain $-n_3$. Although the beam of rays c_3 with direction n_3 intersects these voxels, n_3 is not deleted from the global visibility map of v.

the number of boundary voxels. Figure 3.8 illustrates cases for which the algorithm decides that F is locally inaccessible, *i.e.* at least one boundary voxel with an empty accessibility map will be found. Since the algorithm does not distinguish convex from concave surface intersections, it is always assumed that a voxel v is not accessible with direction $\mathbf{n}$, if $E_{\mathbf{n}}(v)$ contains more than one surface. Figure 3.9 illustrates a violation (concave surface intersection at a voxel v) of accessibility condition 2, and a case (convex surface intersection at a voxel v) that does not violate this condition. But in the latter case the algorithm decides that F is locally inaccessible.

Algorithm 3.4 Algorithm to compute the accessibility maps of the boundary voxels

Initialize for each boundary voxel v of F', $A'(v)$ with $\emptyset$. For each boundary voxel v of F' and for each direction $\mathbf{n} \in V'_g(v)$ check the following three conditions:

1. All approximated global visibility maps of the voxels in $E_n(\mathbf{v})$ contain $\mathbf{n}$, *i.e.*

$$\mathbf{n} \in \bigcap_{w \in E_{\mathbf{n}}(v)} V'_g(w).$$

2. The maximal surface curvature does not exceed $1/\alpha$ at $E_{\mathbf{n}}(v)$.

3. $E_{\mathbf{n}}(v)$ contains only one surface.

Add $\mathbf{n}$ to $A'(v)$, if these conditions are met.

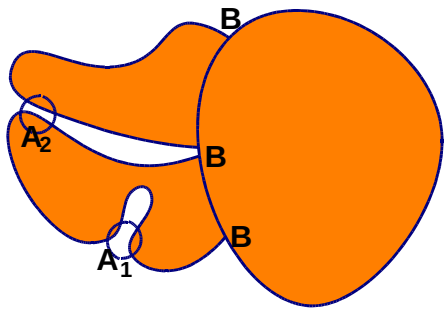


Figure 3.8 If accessibility condition 1 (location A1 and A2) or accessibility condition 2 (location B) of Section 3.2.1 is violated, it is assumed that F is not accessible at these locations.

3.4.4 Selecting a limited number of tool access directions

Let V_A denote the set of boundary voxels with accessibility maps that contain at least one tool access direction. Let D_A denote the directions contained by any of the voxels from V_A. Our goal is to select a limited number of tool access directions from D_A with which all voxels from V_A can be accessed.

Selecting a minimal number N_A of tool access directions is NP-hard (Tang et al. 1992, Chen et al. 1993). N_A can be found by testing V_A's accessibility for all subsets from D_A with n elements. This exhaustive search process would be started with $n = 1$ and halted if a subset of n directions is found with which all voxels of V_A are accessible. In this case a minimal set of tool access directions is found. Otherwise n is increased with one and the process is repeated until a set of directions is found with which V_A is accessible and N_A

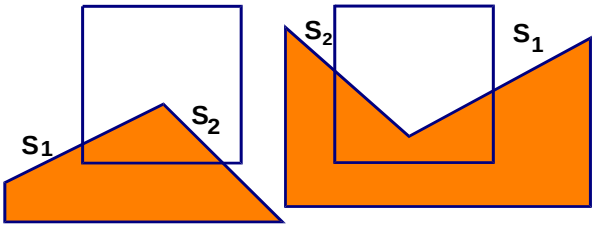


Figure 3.9 The visibility map of a voxel cannot distinguish convex from concave surface intersections, because this map contains only the directions with which the surfaces in that voxel are visible.

---

is set to n. Hence, for large values of N_A, it is impossible to compute a minimal number of tool access directions within a reasonable running time.

Therefore, we implement an algorithm that first tries to determine the minimal number of tool access directions N_A by exhaustive search until $n = n_{max}$, where n_{max} denotes a small number. If all sets containing n_{max} tool directions have been generated without finding a set of directions with which V_A is accessible, then as an intermediate result the set with which the largest number of voxels from V_A can be accessed, is selected. This set is extended using the 'greedy' approach from (Tang et al. 1992, Chen et al. 1993, Janardan & Woo 1997) to approximate the minimal number of set-ups within a logarithmic factor. The direction that extends the accessible part of V_A with the maximum number of voxels is selected. This process is repeated until all voxels from V_A are accessible.

In the implementation of our sculpturing method the computing time as well as the machining time grows with the number of tool access directions. Therefore, we offer the user a trade-off between the accessibility of the boundary voxels and the number of selected tool access directions. If the user specifies a percentage of the voxels from V_A, that should be at least accessible, less tool access directions are selected.

3.5 Extending tool access directions to tool access orientations

To apply machining with a fixed tool orientation as described in Chapter 4 each tool direction has to be extended to a tool orientation. This can be done as follows. In Chapter 2 we have attached to the tool holder a coordinate system, such that the tool axis coincides with the **z**-axis of that coordinate system. The **x**-axis and the **y**-axis of this coordinate system still can rotate around the **z**-axis. To obtain a fixed orientation a

twist angle around the **z**-axis has to be selected that specifies the rotation of the **x**-axis and the **y**-axis around the **z**-axis. Each of the computed tool access directions has to be extended to a tool access orientation in this way, taking into account the limitations of the device to position the tool holder. E.g. in 5-axis milling there exists only one way to extend these tool access direction, because the tool can be positioned only with 5 degrees of freedom. If in 6-axis milling no constraints have to be taken into account, then a tool orientation can be selected such that the accessibility of the CAD shape model is optimal. Chapter 5 contains a description of the extension of the tool access directions taken into account the limitations of our Sculpturing Robot system.

3.6 Conclusions

We have presented a method to extract a limited number of tool access orientations from the shape F of an initial free-form object. First, F's shape, given by a set of B-spline surfaces, is transformed into a voxel map. We allow that the B-spline surface data are as inaccurate as the user-specified geometric accuracy of the initial free-form object. Each voxel is classified either as an exterior, boundary or interior voxel. Next, at each boundary voxel v a number of global tool access directions to access $\partial F \cap v$ is computed. From these tool access directions a limited number of tool access directions are extracted, with which all boundary voxels can be accessed. Finally, each of these tool access directions is extended to a tool access orientation.

The face octrees described in (Brunet, Navazo & Vinacua 1993) may provide a good alternative to represent the spatial coherence of B-spline surfaces in an explicit way, because the size of the face nodes is based on the local surface curvature. With this data structure also concave and convex surface intersections can be distinguished.

Another alternative may be provided by converting initial shape models, that are roughly specified by a B-spline surface geometry, into a voxel map with a relatively low resolution. For each voxel v from this voxel map a normal vector can be estimated from the geometry of v and the voxels in its neighborhood as described in (Yagel, Cohen & Kaufman 1992). Because the number of voxels is larger than with our scheme, computing a global V-map from the local V-map would though require more computing time.

4

Description of sculpturing using spatial planning and Minkowski operations

Path planning for sculpturing a shape involves the description of the erosion of a stock of material by the milling tool. During this process we must avoid interference between the tool holder and the stock, and between the tool and the shape model to be obtained. We show in this chapter, that for a fixed tool orientation, spatial planning and Minkowski operations can be applied to state the sculpturing problem, and to describe, implement and compare NC machining methods. In spatial planning an object is placed or moved among one or more obstacles without interfering them. The object is shrunk to a point $\mathbf{o}$ and the obstacles are dilated with the object. The dilated obstacles then represent the forbidden region as the complement of the free space in which $\mathbf{o}$ can be positioned. Interference avoidance for NC machining involves both a tool holder spatial planning problem and a tool spatial planning problem. In the case of the tool holder spatial planning problem, the tool holder is the object and the stock is the single obstacle. In the case of the tool spatial planning problem, the tool is the object and the shape to be obtained is the single obstacle. In both cases the tool holder and the tool are shrunk to the same point $\mathbf{o}$. Then the forbidden region for positioning $\mathbf{o}$ is the union of the dilated stock and the dilated model and the tool path is extracted from the boundary between the forbidden region and the free space.

In the first part of this chapter we give an overview of related work, we formally describe Minkowski operations on sets and spatial planning, and we describe sculpturing using Minkowski operations on sets. In the second part of this chapter we apply the Minkowski formalism to machining methods based on height grid maps. To this end we formally describe Minkowski operations on these height grid maps. Next, we use Minkowski operations on these height grid maps to describe interference-free machining strategies.

4.1 Previous work

Researchers in the field of mathematical morphology, e.g. (Serra 1982, Giardina & Dougherty 1988), have studied the Minkowski addition and subtraction extensively. In this field, image-processing operations are defined by the Minkowski addition and subtraction as dilations and erosions of sets in 2-dimensional space, respectively. Although mathematical morphology has initially been applied to digital image processing, the Minkowski addition and subtraction are defined generally for n-dimensional space. Therefore, with the Minkowski operations also volume processing can be described. For example, (Menon, Marisa & Zagajac 1994) describe blending of solids by using Minkowski operations. Minkowski operations have also been applied in spatial planning to compute the free space for placing an object amidst a set of obstacles (see e.g. the paper by (Lozano-Pérez 1983), that by (Ghosh 1990) or the book by Latombe (Latombe 1991)). The object, which is modelled as a subset of an n-dimensional space, is allowed to translate freely without rotation. A reference point is attached to the object. The region in which the reference point can move without interference with one of the obstacles is called the free space. If the obstacles are dilated by the object and the object is shrunk to its reference point, the free space consists of the points for which the shrunken object does not intersect the dilated objects, *i.e.* the complement of the union of the dilated obstacles constitutes the free space. (Jonker & Vermeij 1995) apply Minkowski operations to real-time robot path planning using a massively parallel implementation.

A lot of machining methods employ grid maps (often called Z-maps) as the data structure to represent shapes. These methods are based on discretizing a planar area, say a region in the plane perpendicular to the tool approach orientation, into a regular 2D array of squares, called grid elements. One grid map is used to record a height field of the model surface by storing one height value per grid element. Other grid maps are used to store height fields of the stock, the tool and the tool holder. From these height fields, tool paths can be computed. The simplicity of grid methods is very appealing but the major deficiency of this approach is the low accuracy that can be expected. A grid of size $10^4 \times 10^4$ can yield an accuracy of 0.1 mm in a 1 m $\times$ 1 m machining working space. However, this example requires 10^8 cells, a large amount of memory space even for top-of-the-line computing systems. This accuracy is not sufficient for high-precision metal cutting, but sufficient for fast milling of foam shape models.

Each of the following authors describes aspects of the relation between NC machining, image-processing and Minkowski operations. As far as we know no NC machining strategies that avoid tool interference as well as tool holder interference using Minkowski operations, have been described. (Saito & Takahashi 1991) focus on the generation of NC machining paths using image-processing operations, but they do not use Minkowski operations to describe their NC tool path generation algorithm. (Vepsäläinen 1990) describes in a short paper tool interference avoidance for NC machining. He discusses the principle of describing image-processing operations for NC tool path generation; however he does not describe machining strategies and does not discuss tool holder interference. (Kaul 1992) provides a survey of applications of the Minkowski addition. For the NC tool interference problem he generates the tool path from the free space boundary, which

is computed by the Minkowski addition of the surface to be machined and a sphere, that models the ball-end of a milling tool. (Head, Kedem & Prisant 1994) describe a Minkowski procedure to compute a molecular contact surface via one Minkowski addition and one Minkowski subtraction. (Menon et al. 1994) describe algorithms and a ray-rep representation with which the Minkowski addition and subtraction can be implemented efficiently for volume processing operations like blending solids.

In a recent paper (Choi, Kim & Jerard 1997) propose a formal approach to 3-axis NC tool path generation. Taking into account tool interference they distinguish the following three disjoint spaces and the two boundaries between these spaces as configuration-space elements:

- A free space in which the tool reference point is positioned such that the tool does not interfere with the stock.
- A machining space in which the tool reference point is positioned such that the tool interferes with the stock but does not interfere with the shape to be sculptured.
- A gouging space in which the tool reference point is positioned such that tool interferes with the shape to be sculptured.
- The boundary between the free space and the machining space.
- The boundary between the machining space and the gouging space.

From a given configuration a specific pattern of machining operations can be derived and a number of these patterns are employed to machine a shape. (Choi et al. 1997) recommend future research on the mathematical foundation of their approach and on its implementation by a grid map data structure. Minkowski operations are not mentioned in their paper. In the remainder of this chapter we show that Minkowski operations provide a mathematical foundation to specify the sculpturing problem and to describe NC machining methods using grid maps.

4.2 Mathematical background

4.2.1 Minkowski operations

We have found in the literature several conventions to define Minkowski operations. Chapter 7 of the book by (Ritter & Wilson 1996) provides a short introduction to Minkowski operations and lists a number of conventions used in the literature. We adopt the convention used by (Serra 1982) for the definition of the Minkowski addition and subtraction. Our definition of dilation and erosion follows (Kaul 1992) instead of (Serra 1982), because Kaul's definitions are more appropriate for describing machining.

The Minkowski addition of two sets A and B, each consisting of vectors in $\mathbb{R}^n$, the Euclidean space of dimension n, is constructed by translating A by each element of B and taking the union of all the resulting translates, as illustrated in Figure 4.1. Intuitively,

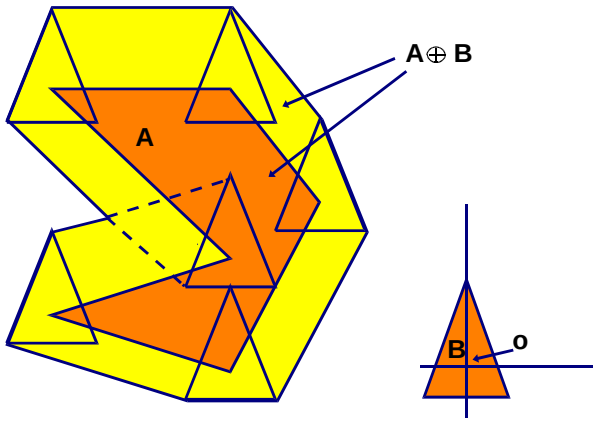


Figure 4.1 The Minkowski addition $A \oplus B$ is constructed by translating A by each element of B and taking the union of all the resulting translates. The triangles denote translates of the vertices of A.

---

the Minkowski addition can be considered as a dilation process where one set is expanded by the other. In more formal terms, the Minkowski addition is defined as

$$A \oplus B = \cup_{\mathbf{b} \in B} A + \mathbf{b},$$

where $A + \mathbf{b} = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A\}$ denotes a translation of A by a vector $\mathbf{b}$. The Minkowski addition is commutative, *i.e.*

$$A \oplus B = B \oplus A.$$

The Minkowski subtraction of two sets A and B is constructed by translating A by each element of B and taking the intersection of all the resulting translates as illustrated in Figure 4.2. Intuitively, the Minkowski subtraction can be considered as an erosion process where one set is eroded by the other. In more formal terms, the Minkowski subtraction is defined as

$$A \ominus B = \cap_{\mathbf{b} \in B} A + \mathbf{b}.$$

The Minkowski addition and subtraction are dual operations, *i.e.* dilating (eroding) a set A by a set B is equivalent with eroding (dilating) its complement A^c by B, where $A^c = I\!R^n \backslash A$. In more formal terms,

$$A \oplus B = (A^c \ominus B)^c$$

and

$$A \ominus B = (A^c \oplus B)^c.$$

If B models a milling tool, in our convention the erosion of A by B, *i.e.* $A \ominus B$ is the volume that is left of A after the origin of the coordinate system, in which B is defined, has been positioned everywhere outside A. (In mathematical morphology the erosion with a volume B corresponds with shape removal by $-B$, which is not an appropriate model for machining.)

Next, we use the Minkowski operations to define the spatial planning problem.

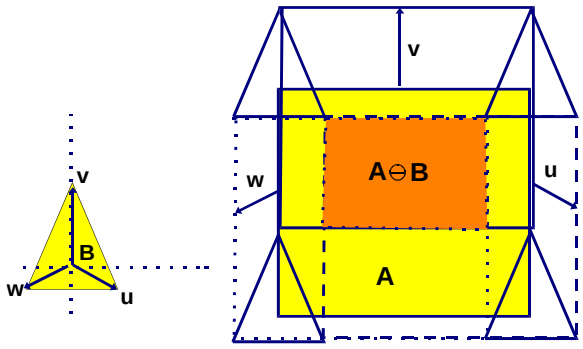


Figure 4.2 The Minkowski subtraction $A \ominus B$ is constructed by translating A by each element of B and taking the intersection of all the resulting translates.

4.2.2 Spatial planning

Below, we give a short description of the spatial planning problem. For an introduction on spatial planning we refer the reader to (Lozano-Pérez 1983, Ghosh 1990, Latombe 1991).

Let an object be given as a subset V of an n-dimensional space $\mathbb{R}^n$. Let one or more obstacles be given as a subset O of $\mathbb{R}^n$. Assume that a Cartesian coordinate system $\mathcal{V}$ is attached to V and a Cartesian coordinate system $\mathcal{O}$ is attached to O. The origin of $\mathcal{V}$ is a reference point for the object V. Assume that V is a free-translating object in $\mathcal{O}$, *i.e.* $\mathcal{V}$ can translate but cannot rotate in $\mathcal{O}$. Furthermore, assume that the orientations of the x, y and z-axes of $\mathcal{O}$ are identical to those of $\mathcal{V}$.

O interferes with V, if and only if the origin of $\mathcal{V}$ is positioned at a point $\mathbf{p}$ for which O interferes with $V + \mathbf{p}$. Hence, the forbidden region for position the origin of $\mathcal{V}$ is

$$\begin{aligned} & \{\mathbf{p} \mid O \cap (V + \mathbf{p}) \neq \emptyset\} \\ = \; & \{\mathbf{p} \mid \exists \mathbf{o} \in O, \mathbf{v} \in V : \mathbf{o} = \mathbf{v} + \mathbf{p}\} \\ = \; & \{\mathbf{o} - \mathbf{v} \mid \mathbf{o} \in O \wedge \mathbf{v} \in V\} \\ = \; & O \oplus (-V) \end{aligned}$$

Hence, $O \oplus (-V)$ is the forbidden region and $(O \oplus (-V))^c$ the free space for positioning the origin of $\mathcal{V}$ relative to $\mathcal{O}$ (see also Figure 4.3).

4.3 Description of the sculpturing problem

In this section we describe the sculpturing process, the computation of the free space for positioning the tool, accounting for tool interference and tool holder interference, and the computation of the stock-in-progress, that would be left if the tool is positioned

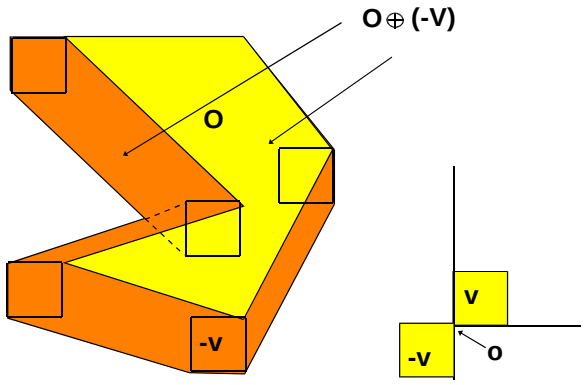


Figure 4.3 V and O intersect, if $\mathbf{o} \in O \oplus (-V)$.

everywhere in the free space. Note that in our approach the free space includes the machining space described by (Choi et al. 1997). An NC machining process obtains a shape model R of a shape F, which is given as a solid model, by removing material from an initial stock that encloses F. Grid height methods are often applied for rough NC machining as described by You and Chu (You & Chu 1995). Requirement 7 from Chapter 1 states that each milling process must be performed in such a way that a subsequent, more accurate machining process can be made. Therefore, we require that the tool never penetrates F during machining, although we allow that the tool touches the boundary of F. So, R will enclose the interior of F. Hence, interference between the tool and the interior of F is not allowed. The stock S is a volume that shrinks from the initial stock to the shape model R due to the milling tool T. Also we require that the tool holder never collides with the stock-in-progress during machining. We allow the tool holder to touch the boundary of S, but interference between the tool holder and the interior of S is not allowed.

Suppose that, in a particular process, the tool path follows the boundary of the free space. If the tool reference point $\mathbf{o}$ is positioned on this boundary, the tool might touch F, *i.e.* it can intersect the boundary δF of the free-form shape F, but it cannot intersect the interior F^o of F. Also the tool holder H can intersect the stock boundary δS, but it cannot intersect its interior S^o.

Typically, an NC machining process consists of a number of stages. At each stage a milling path is generated which consists of one or more movements of the milling tool to shrink the stock-in-progress. One machining stage can be described as follows.

- Compute the forbidden space for positioning the milling tool.
- Generate a milling path outside the forbidden space.
- Update the stock-in-progress.

Below we describe interference-free NC machining as a spatial planning problem consisting of a tool interference avoidance problem and a tool holder interference avoidance problem. These two problems can be solved separately, because the forbidden region for positioning the milling tool is obtained as the union of the forbidden regions of both interference avoidance problems.

Problem 4.1 Interference avoidance for NC machining
Let a shape F and a stock-in-progress S be given in the same Cartesian coordinate system $\mathcal{S}$. Let the tool T and its tool holder H be given in the Cartesian coordinate system $\mathcal{T}$. Assume that the tool reference point of T coincides with the origin $\mathbf{o}$ of $\mathcal{T}$. Assume that $T \cup H$ is a free-translating object in $\mathcal{S}$, i.e. $\mathcal{T}$ can translate but $\mathcal{T}$ cannot rotate in $\mathcal{S}$. Further, assume that the orientations of the x, y and z-axes of $\mathcal{S}$ are identical to those of $\mathcal{T}$. Compute the forbidden region D for positioning the milling tool, such that both tool interference and tool holder interference are avoided.

The tool interference avoidance problem is now a spatial planning problem, where the tool T is the object that cannot intersect the interior F^o of the shape F, which is the single obstacle. The tool holder interference avoidance problem is a similar spatial planning problem, where the tool holder H is the object that cannot intersect the interior S^o of the stock-in-progress S, which is the single obstacle. Positioning the origin $\mathbf{o}$ of $\mathcal{T}$ at a point $\mathbf{p}$ in $\mathcal{S}$ is allowed if and only if $\mathbf{p}$ is outside the union of the forbidden regions of both interference avoidance problems. The forbidden region for the tool interference problem is $F^o \oplus (-T)$ and the forbidden region for the tool holder interference problem is $S^o \oplus (-H)$. Hence, the forbidden region D for positioning the milling tool is $(F^o \oplus (-T)) \cup (S^o \oplus (-H))$ and D^c is the free space.

In general, less material will be removed if the origin $\mathbf{o}$ of $\mathcal{T}$ follows some finite milling path outside the forbidden region instead of being positioned everywhere outside the forbidden region, which is impossible in practice. However, we will show later in this chapter that for representations based on grid maps the following assumption is valid.

Assumption 4.1 *The free space boundary contains a machining path with finite length for positioning the origin $\mathbf{o}$ of $\mathcal{T}$ that removes the same material as would have been removed if $\mathbf{o}$ is positioned everywhere outside the forbidden region.*

Next, we describe updating the stock-in-progress S as a volume removal problem.

Problem 4.2 *Assume that the origin $\mathbf{o}$ of $\mathcal{T}$ is positioned at every point $\mathbf{p}$ inside the free space D^c. Determine the volume that will be left in the stock-in-progress S.*

The volume swept by T is

$$\cup_{\mathbf{p} \in D^c} T + \mathbf{p} = T \oplus D^c = D^c \oplus T.$$

The volume not swept by T is $(D^c \oplus T)^c = D \ominus T$. Hence, $S \cap (D \ominus T)$ is left of the stock-in-progress S. If no tool holder collision avoidance is necessary, *i.e.* if $(S^o \oplus (-H)) \subseteq$

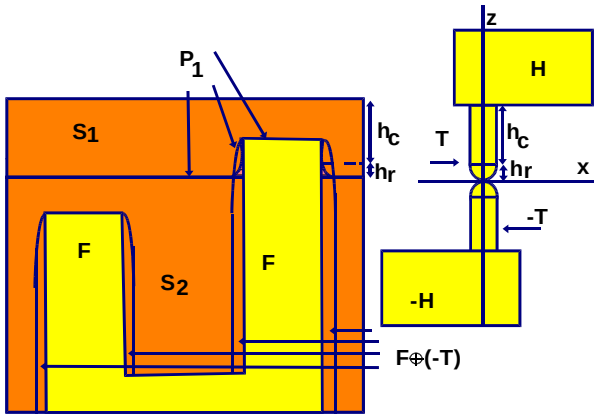


Figure 4.4 First step of the milling process, where P_1 is the boundary of $D_1 = (F^o \oplus (-T)) \cup (S_1^o \oplus (-H))$ and $S_2 = S_1 \cap (D_1 \ominus T)$.

$(F^o \oplus (-T))$, then $(F^o \oplus (-T)) \ominus T$ will be obtained by the milling process. In the field of mathematical morphology $(F^o \oplus (-T)) \ominus T$ denotes the closing of F^o with the structuring element $-T$, *i.e.* the closing of F^o is obtained by first dilating F^o with $-T$ and then eroding the result with $-T$, using $A \ominus (-B)$ instead of $A \ominus B$ to denote the erosion of a set A by a structuring element B.

The theory presented above, can be applied straightforwardly to generate interference-free tool paths iteratively, such that per iteration a milling path in the free space boundary is generated and the stock-in-progress is updated. This is expressed by Algorithm 4.1.

Algorithm 4.1 Free space boundary algorithm

Let $S_1 \supseteq F$ denote the initial stock-in-progress. Let S_k denote the stock-in-progress before machining stage k. Let D_k denote the forbidden region for positioning the milling tool in machining stage k. Let R denote the shape to be obtained by the milling process. Let n denote the number of milling stages.

$k := 1$;
repeat
  $D_k := (F^o \oplus (-T)) \cup (S_k^o \oplus (-H))$;
  Generate a finite machining path P_k in the free space boundary D_k^c for positioning **o** of $\mathcal{T}$ that removes the same material as would have been removed if **o** of $\mathcal{T}$ is positioned everywhere outside D_k;
  $S_{k+1} := S_k \cap (D_k \ominus T)$;
  $k := k + 1$;
until $(S_{k-1} = S_k)$
$n := k - 1$; $R := S_n$; $D := D_n$;

Figure 4.4 illustrates the first step of this algorithm.

Algorithm 4.1 generates machining paths that follow the free-space boundary repeat-

edly. Hence, as much material as possible is removed. In NC machining often paths are generated that remove material slice by slice. Next, we present an algorithm that extracts from the final free space boundary δD^c paths for machining in a sliced fashion. In the description of the algorithm we restrict ourselves to cutting tools that consist of a cylindrical part and a rotation-symmetric part. In the case of a ball-end tool the remaining part is a hemisphere with the same radius as the cylindrical part and in the case of a flat-end tool the remaining part is empty.

We assume that the tool consists of a cylindrical part with height h_c and a rotation-symmetric remaining part with height h_r. Without loss of generality, let the tool direction be in the $-z$ direction and the tooltip be positioned at the origin $\mathbf{o}$ of $\mathcal{T}$ as indicated in Figure 4.4. Further we assume that T rotates around the z-axis of $\mathcal{T}$. Let the boundary of the forbidden region be given as a height function d, with domain $D_{xy} = \{(x, y) \mid \exists z : (x, y, z) \in D\}$ such that for all $(x, y) \in D_{xy}$ it holds that $d(x, y) = sup\{z \mid (x, y, z) \in D\}$. Then with Algorithm 4.2 machining paths, that remove material from the stock-in-progress slice by slice, are computed.

Algorithm 4.2 Algorithm to compute the milling paths slice by slice

Compute D with Algorithm 4.1;
s_{max}:= the maximal height of S_0;
s_{min}:= the minimal height of S_0;
$to := s_{max}$; $from := to - h_c$;
$l := \lceil (s_{max} - (s_{min} + h_r))/h_c) \rceil$;
for $k := 1$ **to** l **do**
  (* *S and R are identical for* $z > to$ *)
  (* *Obtain the final result between* $from$ *and* to *)
  Generate a machining path P_k that follows
  $\{(x, y, z) \mid (x, y) \in D_{xy} \wedge d(x, y) < to \wedge z = max(from - h_r, d(x, y))\}$;
  (* *S and R are identical for* $z > from$ *)*;
  $to := from$; $from := to - h_c$;
end for

Each machining path P_k removes a slice of material by partially following a plane parallel to the xy-plane at height $from - h_r$ and by partially following the free space boundary between $from$ and to. If the kth slice is being machined, the stock-in-progress S is always identical to R above the height to and upon completion of milling this slice, S is identical to R above the height $from$. Since the tool holder H can hit S only above the kth slice, where S is identical with R, interference between S and H is avoided if the machining path follows or is above the free space boundary δD^c.

Next we define Minkowski operations on grid maps and we describe the implementation of the two interference-free NC machining algorithms using grid maps.

4.4 Minkowski operations on numerical functions

In grey-scale morphology (Giardina & Dougherty 1988, Serra 1982) the definitions of Minkowski operations on sets are modified to real-valued functions defined on finite subsets of $\mathbb{R}^n$, the Euclidean space of dimension n, and on $\mathbb{Z}^n$, the Euclidean grid of dimension n. These functions are called numerical functions. We will use numerical functions as a formalism to describe grid height fields. Using numerical functions on $\mathbb{Z}^2$ and Minkowski operations we will describe the implementation of the two machining algorithms that we have presented.

Let $D_f \subset \mathbb{R}^2$ denote the domain of the function f. Let the translation of f by a point (s,t) be denoted by $f_{s,t}$. $f_{s,t}$ is defined as

$$D_{f_{s,t}} = \{(x,y) \mid (x-s, y-t) \in D_f\}$$

and for all $(x,y) \in D_{f_{s,t}}$

$$f_{s,t}(x,y) = f(x-s, y-t).$$

Given a set S, the reflected set $\hat{S} = \{(x,y,-z) \mid (x,y,z) \in S\}$ is symmetrical to S with respect to the xy-plane.

The maximum of a finite collection F of numerical functions is defined on $\bigcup_{f\in F} D_f$ by $MAX(F)(x,y) = \max\{f(x,y) \mid f \in F \wedge (x,y) \in D_f\}$.

The minimum of a finite collection F of numerical functions is defined on $\bigcap_{f\in F} D_f$ by $MIN(F)(x,y) = \min\{f(x,y) \mid f \in F\}$.

The volume represented by a numerical function f on $\mathbb{R}^2$ is called its umbra $U(f)$. This is the set of all points that lie below f, *i.e.*

$$U(f) = \{(x,y,z) \mid (x,y) \in D_f \wedge z \leq f(x,y)\}.$$

In mathematical morphology the Minkowski addition $f \oplus g$ of two numerical functions f and g on $\mathbb{R}^2$ is defined as a dilation of f with g such that $U(f \oplus g) = U(f) \oplus U(g)$, *i.e.*

$$D_{f\oplus g} = D_f \oplus D_g$$

and for all $(x,y) \in D_{f\oplus g}$

$$(f \oplus g)(x,y) = \sup_{(s,t)\in D_g} \{f_{s,t}(x,y) + g(s,t)\}.$$

Similarly, the Minkowski subtraction $f \ominus g$ of two numerical functions is defined as an erosion of f with g such that $U(f \ominus g) = U(f) \ominus \hat{U}(g)$, i.e,

$$D_{f\ominus g} = D_f \ominus D_g$$

and for all $(x,y) \in D_{f\ominus g}$

$$(f \ominus g)(x,y) = \inf_{(s,t)\in D_g} \{f_{s,t}(x,y) - g(s,t)\}.$$

Note that for numerical functions on $\mathbb{Z}^2$ with finite domains, *sup* and *inf* are identical to *max* and *min*, respectively.

Given two numerical function f and g on $\mathbb{Z}^2$ with finite domains containing m and n elements, respectively, $f \oplus g$ can be computed straightforwardly in $O(mn)$ time. Also $f \ominus g$ can be computed straightforwardly in $O(mn)$ time. In the field of mathematical morphology there have been a number of publications (Young, Peverini, Verbeek & van Otterloo 1981, van Vliet & Verwer 1988, Vincent 1991) that present much faster algorithms based on contour processing to compute the Minkowski addition and subtraction.

Given a grid size d, we associate with a numerical function f on $\mathbb{Z}^2$ a height map as follows. Let the mapping M from numerical functions on $\mathbb{Z}^2$ to numerical functions on $\mathbb{R}^2$ be given by

$$M(f)(x,y) = f(\lfloor x/d \rfloor, \lfloor y/d \rfloor),$$

where $D_{M(f)} =$

$$\{(x,y) \in \mathbb{R}^2 \mid \exists\,(i,j) \in D_f : id \leq x < (i+1)d \wedge jd \leq y < (j+1)d\}.$$

With a numerical function f on $\mathbb{Z}^2$ the volume $U(M(f))$ is associated.

Now f represents the boundary of the volume $U(M(f))$ as a height field. We will use numerical functions to represent the stock-in-progress, the shape to be machined, the boundary of the forbidden region, the tool and the tool holder. We need a representation for the volume swept as well as for the volume left by the numerical function representing the tool. Also, we need a representation for the volume swept by the numerical function representing the tool holder. The following property describes the volume swept by $U(M(g))$ and the volume left by $\hat{U}(M(g))$ for a numerical function g on $\mathbb{Z}^2$ if the origin of its coordinate system follows the boundary $M(f)$ of another numerical function f on $\mathbb{Z}^2$ in another coordinate system.

Property 4.1 *Let the numerical function f and g be given on $\mathbb{Z}^2$ in the Cartesian coordinate systems $\mathcal{S}$ and $\mathcal{T}$, respectively. Let $Swept(g) = MAX(g, g_{1,0}, g_{0,1}, g_{1,1})$.*
Then,

$$M(f) \oplus M(g) = M(f \oplus Swept(g))$$

and

$$M(f) \ominus M(g) = M(f \ominus Swept(g)).$$

Proof: From the definition of the Minkowski addition follows

$$U(M(f) \oplus M(g)) = U(M(f)) \oplus U(M(g)).$$

If the origin **o** of $\mathcal{T}$ sweeps the square between $(0,0,0)$, $(d,0,0)$, $(0,d,0)$ and $(d,d,0)$ in $\mathcal{S}$, then the volume swept by $U(M(g))$ is $U(M(Swept(g)))$. If **o** is positioned everywhere at the boundary of $(M(f))$ then the volume swept by $U(M(g))$ is $U(M(f \oplus Swept(g)))$. The same volume is swept if **o** is positioned everywhere at $U(M(f))$ instead. Hence,

$$U(M(f)) \oplus U(M(g)) = U(M(f \oplus Swept(g))).$$

From

$$U(M(f) \oplus M(g)) = U(M(f)) \oplus U(M(g)) = U(M(f \oplus Swept(g)))$$

follows

$$M(f) \oplus M(g) = M(f \oplus Swept(g)).$$

From the definition of the Minkowski subtraction follows

$$U(M(f) \ominus M(g)) = U(M(f)) \ominus \hat{U}(M(g)).$$

If the origin $\mathbf{o}$ of $\mathcal{T}$ sweeps the square between $(0,0,0)$, $(d,0,0)$, $(0,d,0)$ and $(d,d,0)$ in $\mathcal{S}$, then the volume swept by $(\hat{U}(M(g))$ is $(\hat{U}(M(Swept(g)))$. If $\mathbf{o}$ is positioned everywhere at the boundary of $M(f)$ then $U(M(f \ominus Swept(g)))$ is not swept by $\hat{U}(M(g))$. Hence,

$$U(M(f)) \ominus \hat{U}(M(g)) = U(M(f \ominus Swept(g))).$$

From

$$U(M(f) \ominus M(g)) = U(M(f)) \ominus \hat{U}(M(g)) = U(M(f \ominus Swept(g)))$$

follows

$$M(f) \ominus M(g) = M(f \ominus Swept(g)).$$

End of proof

If the origin $\mathbf{o}$ of $\mathcal{T}$ is positioned only at the corner points $(0,0,0)$, $(d,0,0)$, $(0,d,0)$ and $(d,d,0)$ of a square in $\mathcal{S}$ then a numerical function on $\mathbb{Z}^2$ sweeps the same volume as when the origin $\mathbf{o}$ of $\mathcal{T}$ were positioned everywhere on the square. Similarly, if $\mathbf{o}$ is positioned everywhere at the boundary of f it is sufficient to position $\mathbf{o}$ for each $(i,j) \in D_f$ at the points $(di, dj, f(i,j))$, $(d(i+1), dj, f(i,j))$, $(di, d(j+1), f(i,j))$ and $(d(i+1), d(j+1), f(i,j))$.

Therefore, we require that for a numerical function f in $\mathcal{S}$, its tool path connects the points from the set

$$\bigcup_{(i,j)\in D_f} \{(di, dj, f(i,j)), (d(i+1), dj, f(i,j)), (di, d(j+1), f(i,j)), (d(i+1), d(j+1), f(i,j))\}$$

using line segments that do not intersect the interior of f's umbra. Hence, Assumption 4.1, which stated that the free space boundary contains a finite machining path, is valid if the numerical function that represents the free space boundary has a finite domain.

Next, we apply this theory to describe the implementation of the two machining algorithms presented before with numerical functions on $\mathbb{Z}^2$ with finite domains. We do not compute tool paths, but we compute numerical functions from which finite tool paths can be extracted as indicated above.

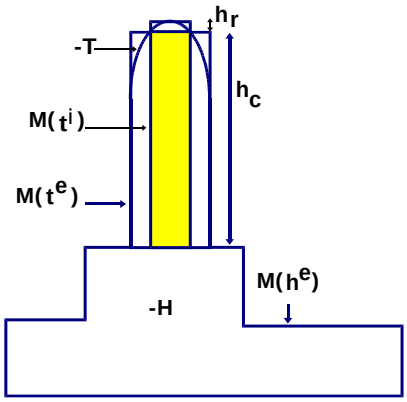


Figure 4.5 The milling tool T is represented by the minimal numerical function t^e such that the umbra of $M(t^e)$ encloses $-T$ and by the maximal numerical function t^i such that the umbra of $M(t^i)$ is enclosed by $-T$. $-H$ is represented by the minimal numerical function h^e such that the umbra of $M(h^e)$ encloses $-H$. $-\min\{t^i(j) \mid j \in D_{t_i}\}$ approximates h_r and, $\max\{h^e(j) \mid j \in D_{h^e}\} - \min\{t^i(j) \mid j \in D_{t_i}\}$ approximates h_c.

4.5 Tool path computation

Given a fixed tool orientation and a grid size d, we will use numerical functions $f : D_f \to \mathbb{R}$, where $D_f \subset \mathbb{Z}^2$ is a finite set, to approximate the stock-in-progress S, the shape F, the tool holder H and the milling tool T. We also use numerical functions on $\mathbb{Z}^2$ to describe the free space boundary from which tool paths are extracted.

Since the obtained shape model should always enclose F, we represent the actual stock-in-progress S by a minimal numerical function s^e such that the umbra of $M(s^e)$ encloses S. Since collision between F and T should be avoided we also represent F and $-T$ by such enclosing numerical functions f^e and t^e. We can choose the maximal value of t^e to be equal to zero. Also collision between S and H should be avoided. Hence, we use a minimal enclosing numerical function h^e to represent $-H$. We assume that the maximal value of h^e is equal to $-(h_r + h_c)$ as indicated in figure 4.5, *i.e.* $h^e(x, y) \leq -(h_r + h_c)$ for all $(x, y) \in D_{t^e}$. Because s^e must be updated such that it encloses the actual stock-in-progress, we need also a maximal numerical function t^i such that the umbra of $M(t^i)$ is enclosed by $-T$. Since T is rotation-symmetric around the z-axis of $\mathcal{T}$, $-t^i$ approximates T and we can write $\hat{T} = -T$. Figure 4.5 illustrates these representations of T and H. Also we use a numerical function p to represent the free space boundary, from which the tool path is extracted. We have shown that the Minkowski addition and subtraction of these functions can be computed straightforwardly.

With these numerical functions the "free space boundary following" machining strategy is described by Algorithm 4.3 which computes the shape r obtained by the machining process and a set of free space boundaries p_i, from which tool paths can be extracted.

Algorithm 4.4 extracts numerical functions q_k from the final free space boundary path

Algorithm 4.3 Algorithm to compute free space boundary following paths

Let n denote the number of milling stages. Let p denote the tool path with which the final result is obtained. Let the upper bound of the initial stock be given by the numerical function s_1^e.

$k := 1$;
(* *Use* $Swept(t^e)$ *to compute* $F \oplus (-T)$ *)
$f^d := f^e \oplus Swept(t^e)$;
repeat
  (* *Use* $Swept(h^e)$ *to avoid interference* *)
  $p_k := MAX(f^d, s_k^e \oplus Swept(h^e))$;
  (* *Use* $Swept(t^i)$ *to update the stock-in-progress* *)
  $s_{k+1}^e := MIN(s_k^e, p_k \ominus Swept(t^k))$;
  $k := k + 1$;
  (* *Stop if no more material can be removed* *)
until $(s_{k-1}^e = s_k^e)$;
(* s_{k-1}^e *is the final result of the machining process* *)
$n := k - 1$; $r := s_n^e$; $p := p_n$;

p. These numerical functions q_k represent sliced tool paths. With these tool paths the same final result r is obtained. In this algorithm we use the following SLI operation on a numerical function on $\mathbb{Z}^2$ to extract sliced tool paths.

$$SLI_{from}^{to}(f)(x,y) = \begin{cases} s_{max} & \text{if } f(x,y) \geq to \\ f(x,y) & \text{if } from \leq f(x,y) < to \\ from & \text{if } f(x,y) < from \end{cases}$$

Algorithm 4.5 computes "sliced" tool paths directly instead of extracting them from the free space boundary. With these tool path surfaces the same final result r is obtained. If a tool path surface p_i is contained in a slice, the tool holder can hit the stock-in-progress only above that slice. Therefore, with Algorithm 4.5 the tool path surfaces are computed efficiently.

The most CPU-intensive part of Algorithm 4.5 is the computation of q by the inner loop

for all $(i,j) \in D_{s^e \oplus t^i}$ **do**
  $q(i,j) := MAX(q, s \oplus h^e)(i,j)$;
end for

where $s = SLICE_{from}^{to}(s^e)$. This computation can be optimized using a Z-pyramid representation (Laur & Hanrahan 1991) for q, s and h^e. The basic idea of the Z-pyramid is to use the original numerical function as finest level of the pyramid and then combine four z values at each level into one z value at the next coarser level by choosing the maximal z value. At the coarsest level of the pyramid there is one single z value which is the maximal value of the numerical function. Let a minimal Z-pyramid be defined by choosing at the next coarser level the minimal Z-value instead of the maximal Z-value.

Algorithm 4.4 Algorithm that extracts sliced tool paths from the final free space boundary path p

Let the upper bound of the initial stock be given by the numerical function s_1^e and its lower bound as a rectangle with a constant minimal height, such the shape F is enclosed by the initial stock.

Compute p with Algorithm 4.3;
s_{min}:=the minimal height of the initial stock;
s_{max}:=the maximal height of s_1^e;
$to := s_{max}$; $from := to - h_c$;
$l := \lceil (s_{max} - (s_{min} + h_r))/h_c \rceil$;
for $k := 1$ **to** l **do**
(* *If we extract a tool path from q_i in the next slice, the tool holder does not collide with the stock-in-progress for $z > to$* *)
(* *extract the tool path surface q_k from p* *)
$q_k := SLI^{to}_{from-h_r}(p)$;
$to := from$; $from := to - h_c$;
end for

In order to facilitate to read the description of a Z-pyramid below we define its numerical functions on the set of integer rectangles
$\{[x1, x2] \times [y1, y2] \mid x1, x2, y1, y2 \in \mathbb{Z} \wedge x1 \leq x2 \wedge y1 \leq y2\}$.

Definition 4.1 *Let, for a numerical function f, its* maximal Z-pyramid *be given by k numerical functions f_i, $i = 1 \cdots k$, where f_1 is a numerical function having one single value which is the maximal value of f and f_k corresponds with f. The Z-pyramid is derived from the original function as follows. Let $V(t, i)$ denote the integer interval starting at t that contains 2^{k-i} elements, i.e. $V(t, i) = [t, t + 2^{k-i} - 1]$.*

f_k is defined as: $f_k([x, x] \times [y, y]) = f(x, y)$ for all $(x, y) \in D_f$.

f_i, $i = 1 \cdots k - 1$ is defined as

$$f_i(V(2^{k-i}x, i) \times V(2^{k-i}y, i)) =$$

$$\max(f_{i+1}(V(2^{k-i}x, i+1) \times V(2^{k-i}y, i+1)),$$
$$f_{i+1}(V(2^{k-i}x + 2^{k-(i+1)}, i+1) \times V(2^{k-i}y, i+1)),$$
$$f_{i+1}(V(2^{k-i}x, i+1) \times V(2^{k-i}y + 2^{k-(i+1)}, i+1)),$$
$$f_{i+1}(V(2^{k-i}x + 2^{k-(i+1)}, i+1) \times V(2^{k-i}y + 2^{k-(i+1)}, i+1)))$$

Minimal Z-pyramids can be defined analogously. Let $<$ denote the usual partial ordering relation for numerical functions on $\mathbb{Z}^2$. Hence, for two numerical functions f_{i-1} and f_i in a maximal Z-pyramid holds $f_i \leq f_{i-1}$ and for two numerical functions f_{i-1} and f_i in a minimal Z-pyramid holds $f_i \geq f_{i-1}$.

Let a Z-pyramid representation for s and h^e be given. We assume that all these Z-pyramids have the same level (this can be achieved by adding levels with the same single

Algorithm 4.5 Algorithm to compute sliced tool paths

Let s_{max} denote the maximal height of the initial stock s_0^e, let f_{min} denote the minimal height of f^e. Let $D_{f^e} = D_{s^e}$ and for all $(x,y) \in D_{f^e}$ let $f^e(x,y) \leq s_{max}$. Let $SLICE_{from}^{to}(f)(x,y) = f(x,y)$, if $from \leq f(x,y) \leq to$. Otherwise let $SLICE_{from}^{to}(f)(x,y)$ be undefined.

k:=0; s^e:="the initial stock"; $f^d := f^e \oplus t^e$;
(* *l denotes the number of slices* *)
$l := \lceil (s_{max} - (f_{min} + h_r))/h_c \rceil$;
(* *compute the boundaries of the first slice and compute* $F \oplus (-T)$ *)
$to := s_{max}$; $from := to - h_c$; $q := f^d$;
(* *the top slice can be milled without any tool holder collision check* *)
(* *extract the tool path surface* p_1 *from* $q = f^d$ *)
$p_1 := SLI_{from-h_r}^{to}(q)$;
(* *update the stock-in-progress* *)
$s^e := MIN(s^e, p_1 \ominus t^i)$;
for $k := 1$ **to** $l - 1$ **do**
  for all $(i,j) \in D_{s^e \oplus t^i}$ **do**
    $q(i,j) := MAX(q, SLICE_{from}^{to}(s^e) \oplus h^e)(i,j)$;
  end for
  (* *If we follow the tool path surface* q *in the next slice, the tool holder does not collide with the stock-in-progress above that slice* *)
  $to := from$; $from := to - h_c$;
  (* *extract the tool path surface* p_{k+1} *from* q *)
  $p_{i+1} := SLI_{from-h_r}^{to-h_r}(q)$;
  (*update the stock-in-progress *)
  $s^e := MIN(s^e, p_{k+1} \ominus t^i)$;
end for
$r := s^e$;

Z value). From the Z-pyramid definition it follows that $s_i \leq s_{i-1}$ and $h_i^e \leq h_{i-1}^e$. From the definition of Minkowski addition follows $s_i \oplus h_i^e \leq s_{i-1} \oplus h_{i-1}^e$. Let a minimal Z-pyramid be given for q with the same level as the Z-pyramids for s and h_e. From the definition of minimal Z-pyramid follows $q_i \geq q_{i-1}$. Hence, q and its minimal Z-pyramid can be computed by the recursive Algorithm 4.5.

4.6 Conclusions

In this chapter we have shown that Minkowski operations form a powerful tool to specify the sculpturing problem and to describe and compare freeform shape machining algorithms that use fixed tool orientations. We have decomposed the interference avoidance problem into a tool holder spatial problem and a tool spatial problem. In the tool spatial planning problem the tool is the object and the shape to be obtained is the single obstacle.

Algorithm 4.6 Recursive algorithm to compute a minimal Z-pyramid

The minimal Z-pyramid is updated by calling the recursive procedure Update_Q at level 1 with appropriate values for $x1$, $x2$, $y1$ and $y2$.

```
procedure Update_Q(x1,x2,y1,y2,l)
(* l denotes the Z-pyramid level *)
begin
w := s_l ⊕ h_l^e([x1, x2] × [y1, y2]);
if w < q_l([x1, x2] × [y1, y2]) ∧ l < k then
  (* updating q and its Z-pyramid is not necessary *)
  NULL;
else
  (* update q_l and refine the computation of q *)
  q_l([x1, x2] × [y1, y2]) := w;
  (* Let "/" denote integer division, e.g. 5/2=2 *)
  Update_Q(x1,(x1+x2)/2,y1,(y1+y2)/2,l+1);
  Update_Q((x1+x2)/2+1,x2,y1,(y1+y2)/2,l+1);
  Update_Q(x1,(x1+x2)/2,(y1+y2)/2+1,y2,l+1);
  Update_Q((x1+x2)/2+1,x2,(y1+y2)/2+1,y2,l+1);
end if;
end Update_Q;
```

In both problems the tool holder and the tool are shrunk to the same point **o**. We have described the forbidden region for positioning **o** as the union of the dilated stock and the dilated model. Finally, we have described a "free space boundary following" machining strategy and a slicing machining strategy that both extract tool paths from the boundary between the forbidden region and the free space. Both strategies are implemented as part of the software package that is described in Chapter 5. In Chapter 6 we will compare the machining time of both strategies.

5

Software implementation for the sculpturing robot system

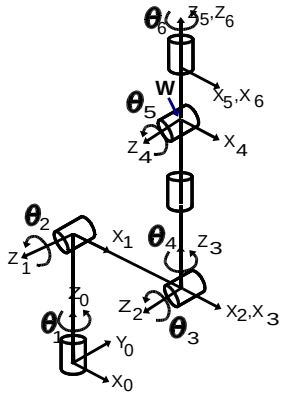
As a testbed for the evaluation of new NC machining methods, a Sculpturing Robot (SR) system has been installed at the Subfaculty of Industrial Design Engineering. The Multiple Access Orientation Sculptor (MAOS) software package has been written to evaluate the methods from Chapter 3 and 4 on the SR system. In this chapter we will describe the SR system and the MAOS software package.

5.1 The sculpturing robot system

In 1988 the Sculpturing Robot Project was initiated for the purpose of automated fabrication of complex shapes (Vergeest 1988). A Sculpturing Robot (SR) system, consisting of a Siemens Manutec R15 robot and a turn table, has been installed. Figure 5.1(a) shows the SR system. The SR system was designed to manufacture foam models of size up to one cubic metre. Therefore, a turn table rotatable around a vertical axis has been positioned in front of the robot. The physical shape model is obtained from a stock of foam, placed on the turn table. The milling process shrinks the stock to a shape model of a free-form object. Two safe configurations have been defined to which the tool can be retracted in the milling process.

Most industrial robots belong to the *wrist-partitioned* class of mechanisms that is sketched in Figure 5.1(b). Also the Manutec R15 robot is an example of a 6R wrist-partitioned manipulator. Such a manipulator consists of 6 links which are connected with 6 revolute joints providing six rotational degrees of freedom. The first three joints are called the *primary joints* and the last three the *minor joints*. The latter orient the tool and the tool holder and have axes that intersect at the wrist point. The first three joints position this wrist point. Manipulators with this design could be said to be composed of a positioning structure followed by an orientating structure or *wrist*. The volume of space that the tool tip can reach is the *workspace* of the manipulator. A tool configuration is specified by both its position and orientation. For 6R wrist-partitioned robots the inverse

(a) The SR system.



(b) 6R wrist-partitioned manipulator. $\mathbf{w}$ indicates the wrist point.

Figure 5.1 The SR system consists of a 6R wrist-partitioned manipulator in conjunction with a turn table on which a stock of foam is mounted.

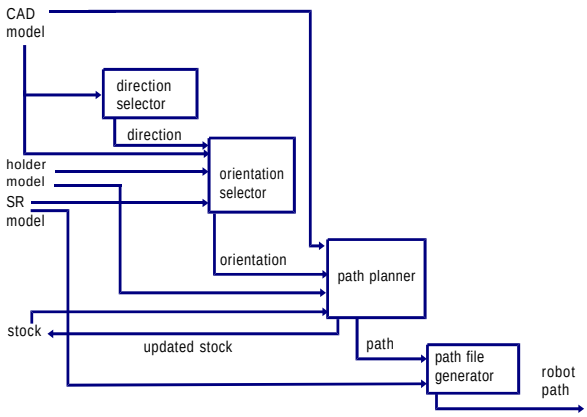


Figure 5.2 MAOS flow chart.

kinematics algorithm from the book by (Craig 1989) can be used to check whether a tool configuration can be reached by the robot and to compute the robot joint angles with which that tool configuration can be reached.

Algorithm 5.1 Round-robin sculpturing algorithm

Let a limited number n of tool access orientations be computed.
Let $0 \leq k < n$.
Let s_k and f_k denote the part of the boundary of the stock-in-progress and the free-form shape, respectively, that can be accessed with the k^{th} tool access orientation.
Let $IndexLastUpdate$ denote the index of the last part of the boundary $s_{IndexLastUpdate}$ that has been updated. The algorithm halts if no more material can be removed, *i.e.* if each $s_k, 0 \leq k < n$ in succession is not updated:

```
Compute the model boundaries f_k, k = 0 ··· n − 1;
Compute the stock boundaries s_k, k = 0 ··· n − 1;
k := 0; IndexLastUpdate = n − 1; Ready := FALSE;
repeat
  Compute the new stock boundary s_new, that can be obtained
  by removing as much material as possible with the k^th orientation
  avoiding tool and tool holder interference;
  if s_new = s_k then
    Ready := (IndexLastUpdate = k);
  else
    Generate sliced tool paths with which s_new is obtained;
    s_k := s_new;
    IndexLastUpdate := k;
    for 0 ≤ l < n ∧ l ≠ k do
      Update s_l taking into account the updated stock grid s_k;
    end for;
  end if;
  k := (k + 1) MOD n;
until Ready;
```

5.2 Overview of the MAOS software

Figure 5.2 presents an overview of the MAOS software modules. Note that the direction selector module and the Minkowski based path planner module do not depend on the SR system model. This model contains kinematic and geometric information about the Sculpturing Robot system[1]. Only the orientation selector and the path file generator take the SR system model into account.

The direction selector module computes a limited number of tool access directions based on the method from Chapter 3. With these directions at least a user-specified percentage of the boundary of the initial shape model can be accessed.

The orientation selector module extends each of these directions into an orientation taking into account the limitations of the Sculpturing Robot system. The orientation selector checks for collision between the tool holder and the turn table, as well as for

[1]For a detailed discussion of the SR system model we refer the reader to (Vergeest 1992).

collision between the tool holder and the initial stock, when the tool holder changes it's orientation. The orientation selector is described in Section 5.4.

The path planner module implements the path generation method based on Minkowski operations as described in Chapter 4. For each tool orientation one or more sculpturing tool paths are generated in a round-robin fashion as described by Algorithm 5.1.

For each tool access orientation the path planner generates one or more "path files". Each path file contains a list of tool positions, that together with the given orientation specifies the tool path.

Finally, the path file generator translates each "path" into a robot path file. In the remaining part of this chapter, these modules are discussed in more detail.

5.3 The direction selector module

Given a shape described by a set of B-spline surfaces and the radius r of a ball-end milling tool, the direction selector computes a limited number of tool access directions. The direction selector[2] implements the method described in Chapter 3. However, the curvature computation has been dropped, because we assume that in practice the number of voxels not accessible due to large surface curvatures, is negligible. To represent a shape, its visibility and its accessibility, the direction selector builds a 3D voxel map. Each voxel v in this map contains a variable that indicates whether a voxel is an exterior, boundary or interior voxel. Each boundary voxel contains a visibility map, a light map, an accessibility map, and a surface identifier, denoting the surface that the boundary voxel intersects. If a boundary voxel intersects more than one surface, this identifier is set to *null*. Recall from Chapter 3, that in this case the voxel is always marked as inaccessible. Visibility maps, light maps and accessibility maps are approximated by finite subsets of the unit sphere as described in Section 3.4.1.

5.4 The orientation selector module

The orientation selector extends a given tool access direction to a tool holder orientation[3]. In this section we only give a summary of the issues that are relevant for the implementation.

A direction is extended to an orientation by including a twist angle value with the direction. In the implementation the twist angle value is selected from a set of values $\{(i/n)2\pi \mid 0 \leq i < n\}$, where the value of n is defined by a parameter. In the tests described in the next chapter we have used 60 as value for n. Using the algorithm from (Craig 1989), the orientation given by a direction $\mathbf{d}$ and the candidate twist angle is converted to a rotation matrix (Rx, Ry, Rz) with $Rz = \mathbf{d}$. Furthermore, for each

[2]For a more detailed discussion of the implementation of the direction selector we refer the reader to (van den Belt & Tangelder 1997*a*).

[3]For an extensive discussion of the implementation of the orientation selector we refer the reader to (van den Belt & Tangelder 1997*b*).

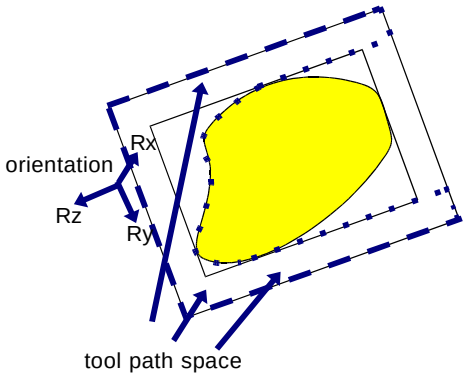


Figure 5.3 In 2D the tool path space is bounded by the three dashed lines and the dotted curve that represents the top of the model to be machined as shown in the figure. In 3D the tool path space is bounded by 5 faces and the surface that represents the top of the model to be machined.

---

orientation the following data is computed:

- A turn table angle value, such that the tool paths will be in the workspace of the robot. The turn table can be rotated around its vertical axis in eight positions with 45 degrees of difference.
- A safe configuration to which the tool can retract without collision.

The orientation selector takes into account the following conditions, based upon the limitations of the SR system, in checking whether a tool orientation is valid or not. For at least one of the eight turn table angle values the following three conditions should be satisfied for the same tool orientation:

- The tool paths should be generated in the workspace of the robot. To check this, a box enclosing the shape model is generated. Each face of this box is perpendicular to one of the column vectors of (Rx, Ry, Rz) as indicated in Figure 5.3. It is assumed that the tool paths are generated in the space between the top of the shape model and the top of the box. A number of sample points are generated. Using the inverse kinematics algorithm for 6R wrist-partitioned robots from (Craig 1989) for each sample point it is tested whether the tool is in the workspace at that sample point.
- If the tool is positioned anywhere in the box B enclosing the shape model, no collision between the tool holder and the turn table should occur. To check this a number of sample points on the faces of the box B are generated. It is assumed that the tool tip is positioned at each sample point and it is tested whether the bounding box B_H of the tool holder is not below the turn table. It is not necessary to sample

the interior of B: if for a tool position $\mathbf{p} \in B$ B_H is below the turn table, then also for a tool position $\mathbf{q}$ on one of B's faces, B_H will be below the turn table.

- It should be possible to retract the tool from the stock-in-progress to at least one of the two safe configurations without collision between the stock and the tool holder. To check this a number of sample points are generated on the retraction path. For each sample point it is tested whether the bounding box of the tool holder and the stock intersect, or not.

One of the orientations that satisfy the conditions above, should be selected. Therefore, for each of the valid orientations the largest free space for positioning the tool is selected. In the free space computation only interference between the tool holder and the shape model is taken into account.

5.5 The path planner

The Minkowski based path planner supports path generation using the method described in Chapter 4. It also contains the grid map data structure to implement numerical functions on $\mathbb{Z}^2$ that have been defined in Chapter 4, and it contains sets of grid maps to represent the object volume including a support structure and the stock-in-progress.

5.5.1 Path generation

The path planner implements Algorithm 4.3 that computes the free space boundary surface as a grid map h. To save programming time we have implemented the Minkowski addition and subtraction straightforwardly instead of using a method based on contour processing.

The Minkowki path planner also implements Algorithm 4.4 that extracts sliced tool paths with a zigzag pattern from the final free space boundary. For each tool access orientation $\mathbf{o} = (Rx, Ry, Rz)$ the tool paths are generated in a Cartesian coordinate system $\mathcal{O}$. Its unit vectors are parallel to Rx, Ry and Rz. There are two options to generate the zigzag paths connecting points (x, y, z) in $\mathcal{O}$. A zigzag path consists of a number of tracks. Either tracks are generated for which the x-coordinate is constant or tracks for which the y-coordinate is constant. In the first case the tracks are extracted from a tool path surface $h(i, j)$, with domain $\{(i, j) \mid i_{min} \leq i \leq i_{max} \wedge j_{min} \leq j \leq j_{max}\}$ as follows.

Let d denote the grid size. Let $\#i = i_{max} - i_{min} + 1$ and let $\#j = j_{max} - j_{min} + 1$. Let the tracks be numbered $j = j_{min} \ldots j_{max}$. The jth track contains the points

$$(di_{min}, dj, h(i_{min}, j)), (d(i_{min} + 1), dj, h(i_{min}, j)),$$

$$(d(i_{min} + 1), dj, h(i_{min} + 1, j)), (d(i_{min} + 2), dj, h(i_{min} + 1, j)),$$

$$\ldots$$

$$(d(i_{max} - 1), dj, h(i_{max}, j)), (di_{max}, dj, h(i_{max}, j)).$$

The tool path consists of straight line segments that connect these points. If $j - j_{min}$ is an even number, the tool follows this track in the same order as its points are listed above. If $j - j_{min}$ is an odd number, the tool follows these points in reverse order.

If $j - j_{min}$ is an even number, a move from the last point in a track $(di_{max}, dj, h(i_{max}, j))$ to the first point $(di_{max}, d(j + 1), h(i_{max}, j + 1))$ in the next track is performed via the point $(di_{max}, d(j + 1), h(i_{max}, j))$.

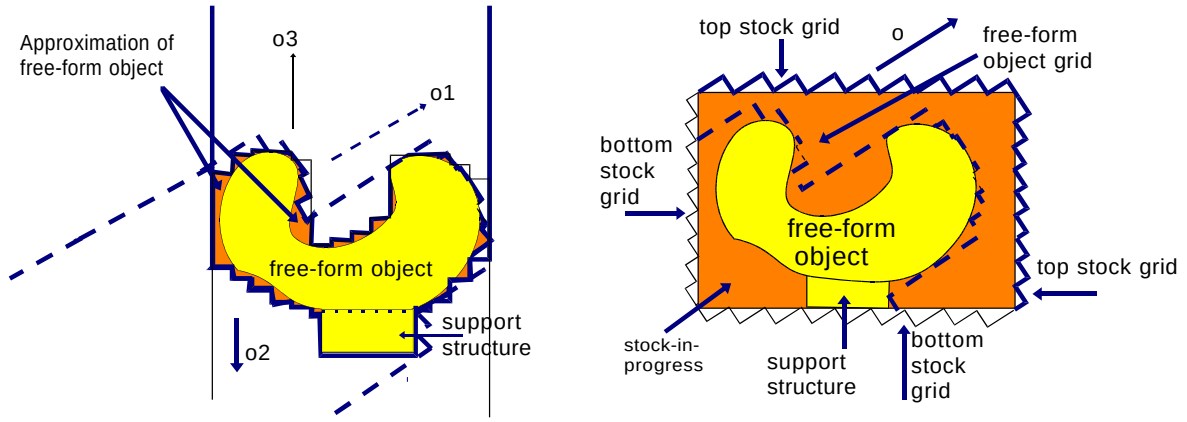
If $j - j_{min}$ is an odd number, a move from the last point in a track $(di_{min}, dj, h(i_{min}, j))$ to the first point $(di_{min}, d(j + 1), h(i_{min}, j + 1))$ in the next track is performed via the point $(di_{min}, d(j + 1), h(i_{min}, j))$.

In the case of a constant y-coordinate, tracks numbered $i = i_{min} \ldots i_{max}$, are generated analogously. Note that the zigzag tool path contains $\#i\#j$-1 line segments with length d and $\#i\#j$-1 other line segments. The software combines line segments if the height in a track does not change for a number of successive line segments, *i.e.* if for some $k > 0$ holds $h(i, j) = h(i + 1, j) = \ldots = (h(i + k, j)$. In this case the 2k-1 line segments between the points $(di, dj, h(i, j))$ and $(d(i + k + 1), dj, h(i + k, j))$ are replaced by a single line segment connecting these two points.

To save machining time, the best out of the two zigzag directions is selected by estimating the machining time for both options. We used the following model to estimate the machining time of a tool path. A tool path consists of a number of line segments. If robot follows a line segment, first it waits D seconds at the start point of the line segment, next accelerates from speed zero with a constant acceleration A until the maximum speed S is reached, decelerates with a constant deceleration A until speed zero is reached at the endpoint of the line segment. It can be derived that following a line segment with length L requires time $D + 2\sqrt{L/A}$, if $L \leq S^2/A$ and time $D + L/S + S/A$ if $L \geq S^2/A$. An estimate of the total machining time is obtained by adding the estimates per line segment. For the sculpturing robot the parameter values are $D = 0.157$ seconds, $S = 1500$ mm/sec and $A = 1960$ mm/sec^2.

Finally, we derive from the tool path pattern a lower bound on the machining time. If we assume that there are no successive line segments connecting points with the same height, following a line segment takes at least D seconds and following a line segment with length d requires time $D + 2\sqrt{d/A}$. Therefore, a lower bound on the machining time for the zigzag tool path is

$$(\#i\#j - 1)(D + 2\sqrt{d/A}) + (\#i\#j - 1)D = 2(\#i\#j - 1)(D + \sqrt{d/A}).$$



(a) The free-form object including the support structure is represented as the volume enclosed by the grid maps of **o1** (dashed lines), **o2** (bold lines) and **o3** (thin lines).

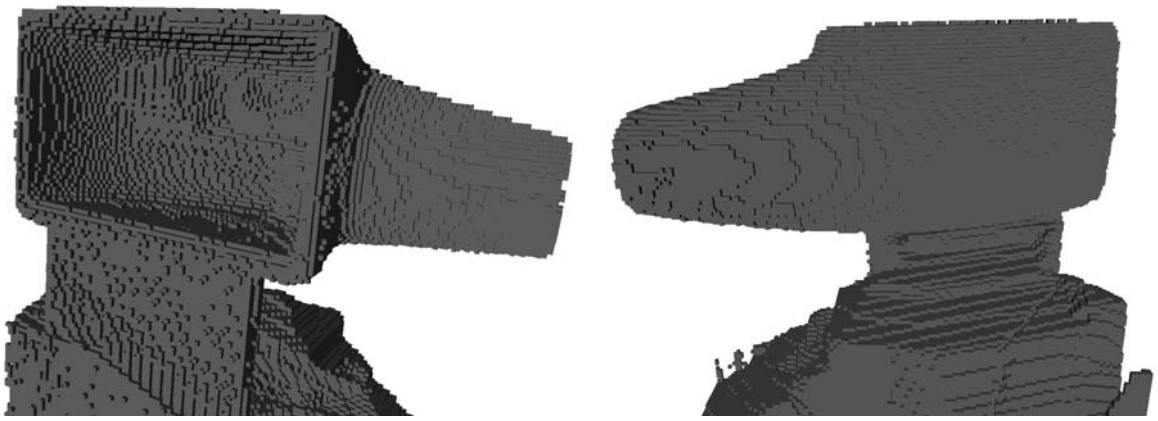
(b) For each tool access orientation **o** three grid maps are built. One grid map (bold lines) represents the top of the free-form object. One grid map (dashed lines) represents the top of the stock-in-progress and one grid map represents the bottom of the stock-in-progress (thin lines).

Figure 5.4 Representation of the stock-in-progress and the free-form object with help of grid maps (2D example).

5.5.2 Data structures for grid maps

Grid maps and sets of grid maps are used to represent volumes. Recall from Chapter 4, that the tool is represented by two numerical functions, one included by the tool, and one containing the tool. Two grid maps are used to implement these two numerical functions. Also, one grid map is used to implement a numerical function representing the tool holder.

Let n tool access orientations be computed by the orientation selector with which a shape model of a free-form object must be obtained from the stock-in-progress. The path planner uses surface sampling to build the n grid maps that enclose the volume of that shape, including the support structure below the shape (see Figure 5.4(a)). For each of the tool access orientations one grid map is generated to represent the boundary of the shape model. For each tool access orientation **o** a grid map f is computed that represents the top of the free-form object relatively to the orientation **o** as illustrated by Figure 5.4(b). We used the robust surface sampling method from (Vergeest et al. 1991) to compute a sufficiently dense set of sample points relatively to **o**'s frame. f's domain D_f consists of all grid elements (i, j) containing at least one sample point. For each grid element $(i, j) \in D_f$ $f(i, j)$ is set to the maximal z-value of the sample points at (i, j). The stock-in-progress is represented by $2n$ grid maps. For each tool access orientation one grid map represents the top of the stock-in-progress and one represents the bottom of the stock-in-progress (see Figure 5.4(b)). Note, that these grid maps are given in the same Cartesian coordinate system as the grid map that represents the boundary of the



(a) View from the front. (b) View from the back.

Figure 5.5 Graphical representation with a 2 mm resolution of a shape model, which can be fabricated with the SR system.

shape model.

The milling paths are generated in a number of milling stages. At each milling stage a constant tool access orientation $\mathbf{o_i}$, $(i = 1, 2, \ldots n)$, is used for which tool paths are generated using the method from Chapter 4. Also, $\mathbf{o_i}$'s top grid map representing the top of the stock-in-progress is updated. Machining of volume that has already been removed with tool access orientation $\mathbf{o1}$, should be avoided. Since the volume between $\mathbf{o_i}$'s bottom grid map and $\mathbf{o_i}$'s updated top grid map encloses the stock-in-progress, all other top stock height grids should also be updated. For each of the orientations $\mathbf{o_j}$, $(j = 1, 2, \ldots n, j \neq i)$, $\mathbf{o_j}$'s top grid map is updated by intersecting it with the volume enclosed by $\mathbf{o_i}$'s bottom and top grid map.

The path planner generates a graphical representation of the shape model to be fabricated. To obtain this graphical representation we have implemented an algorithm that approximates the volume V enclosed by a set of height maps with help of a voxel map. Each voxel is classified as In or Out by testing whether its centre point is an element of V. We have implemented an algorithm that generates the boundary representation of this voxel map in an Inventor file that can be visualized on a Silicon Graphics Workstation. The algorithm writes all the faces that are shared by an In and Out voxel to this file. For an introduction on the Inventor Toolkit we refer the reader to (Wernecke 1994). Figure 5.5 shows a graphical representation of a shape model of a car mirror housing.

5.6 The path file generator

The path file generator implements a robot specific translation of the path data. It transforms a list of path points given in a coordinate system $\mathcal{O}$ to a path file. The path file is a sequence of ASCII encoded command lines, each causing either a linear movement of the milling tool through work space, or a rotation of the turn table or an instruction to start or stop the milling engine[4]. The syntax of the path file is not robot specific and not directly suited to control a robot. The path file is straightforwardly translated into robot specific forms such as (for the SR system) the IRDATA format (VDI 1987)[5].

Also, the path file generator implements a procedure to improve the positioning accuracy of the robot. For sculpturing shapes it is required that the absolute position of tool relative to the turn table is accurately known. A positioning error of more than about 0.5 mm results in unwanted ridges on the shape model at places where milling from one side ends and milling from another side starts. The deviation of a 6R industrial robot from its nominal kinematic model varies typically from 0.5 to 2.0 mm. A significant part of this deviation can be compensated by replacing the nominal kinematic model by a kinematic model derived from the actual placements of the robot rotation axes relative to the turn table (Broek & Vergeest 1996). These actual placements have been obtained from calibration measurements. In addition the calibration resulted in a vector map, over the work space, of the deviation of measured end-effector position from the position predicted by the calibrated kinematic model. With help of this vector map the path file generator corrects the path coordinates. With the calibration and this correction procedure the positioning accuracy has been improved to 0.3 mm for large sized objects (Broek & Vergeest 1996). This accuracy is acceptable for physical shape modeling.

[4]For a description of the syntax of the path file we refer the reader to (Tangelder 1993).

[5]This translation and the partitioning of the path file, which may consists of over 100,000 commands, into portions which are fed into the robot controller, is performed by the SRPOST software (Kooijman 1995).

6

Evaluation of the developed sculpturing method

In the previous chapters we have described a new approach to sculpturing free-form shapes using 6-axis milling. In Chapter 5 the MAOS software, that implements this approach for the Sculpturing Robot system, has been described. This chapter evaluates both the methods from Chapter 3 and 4 separately and the sculpturing strategy implemented by MAOS. First, the validity check of the free-form geometry of the free-form object described in Chapter 3, is illustrated. Next, the free space boundary following strategy and the slicing strategy from Chapter 4 for machining with a single tool access orientation are compared. Then, the test results that have been achieved by MAOS for three free-form shape geometries with different morphological character, are presented and evaluated. Finally, current research is described and extensions to MAOS are recommended.

The tests were performed on a Silicon Graphics Indigo2 workstation with an R10000 processor running at 195 MHZ, rated with 12.3 SPECfp95 and 8.9 SPECint95. This machine can be considered as a medium performance workstation. We used the model from Section 5.5.1 to estimate machining times. We will compare machining times with the lower bound described in Section 5.5.1. Since this lower bound is based on the zigzag tool path pattern from Section 5.5.1, it provides only a yardstick with which the MAOS machining time can be compared. It is not a lower bound for other tool patterns. The lower bound is also not valid for an object, that contains a fragment with a constant grid height. Since in this case successive line segments are replaced by one line segment, the machining time can be lower than the "lower bound".

6.1 The validity check of the free-form object

As described in Chapter 3, MAOS accepts a free-form object if (a) the boundary voxels are connected, (b) there are interior voxels, and (c) for each surface exactly one of its two sides defines the outside of the object. Next, we present two examples of free-form objects that are not valid.

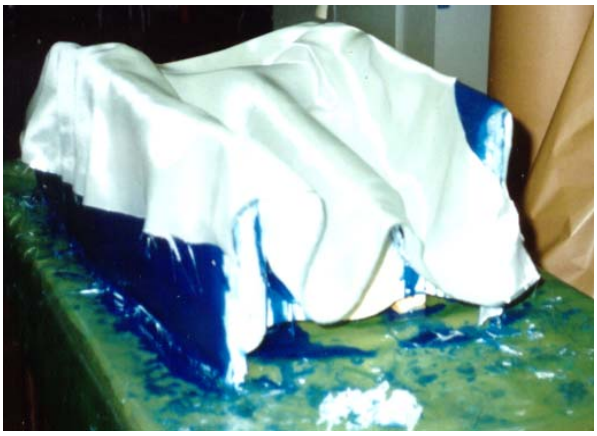
Figure A.1 in Appendix A shows a free-form object of a bear with a displaced nose. This free-form object is not accepted by MAOS, because its boundary consists of two parts. The geometry of the bear is defined by a set of B-spline surfaces. Some of these surfaces intersect each other. For instance, surfaces, that define the legs and the arms of the bear, intersect the surface, that defines the body of the bear. Since MAOS allows that surfaces intersect, this bear will be accepted by MAOS after the nose of the bear is put in its proper place.

The bear object had been modelled erroneously on purpose. We expected, that the free-form object from the next example (Figure A.2 in Appendix A), would be valid. Surprisingly, MAOS did not accept this dust buster object, because in two cases both sides of a surface define the outside of the object. This is detected if MAOS identifies normals that point outwards at both surface sides. In the first case, the surface, that models the body of the dust buster, contains normals pointing outwards in opposite directions. In Figure A.2(a) the body surface normals pointing in one direction are colored green and the normals pointing in the opposite direction are colored red. An investigation of the model showed that the parameterization of this surface is irregular, *i.e.* the isoparametric lines in the u and the v direction are tangent to each other. Therefore the direction of the normal vectors gets inverted if tangent isoparametric lines are passed. From the discussion in Section 3.3, it follows that the other side of the surface is the outside after passing an isoparametric line. Hence, both sides of a surface define the outside of a model. For an introduction on irregular parameterizations we refer to (Farin 1996). In the case of the dust buster the parameterization of the body surface can be made regular by adapting the coordinates of the control points of this surface. In the second case, the green and the orange normals shown in Figure A.2(b) point in opposite directions. In this case an error is detected because the surfaces that should define the dust buster handle do not properly enclose a volume. The free-form object modeling a dust buster will be accepted by MAOS after the parameterization of the surface of the body is made regular and the handle of the dust buster is closed by adding a surface.

6.2 Comparison of free space boundary following strategy and slicing strategy

Before we integrated machining with a fixed tool access orientation in the MAOS software package, we compared the free space boundary following strategy to the slicing strategy. We selected a small and a large test model, each containing one surface that is accessible with one fixed tool orientation. The small test model was a B-spline model of a human face with length 206 mm, width 185 mm and height 116 mm. The large test model was a B-spline model of a seat of a push chair for motor disabled children (see Figure 6.1) with length 948 mm, width 510 mm and height 269 mm. We have simulated the machining process of the human face and the seat of the push chair model. Figure A.3 in Appendix A, that has been generated by MAOS using the Inventor tool kit, provides a graphical presentation of the machining process. Also, using sliced tool paths, we have

(a) Use of the push chair in practice.



(b) The shape model of the push chair seat is strengthened with a polyester fiber structure.

Figure 6.1 A hand-made prototype of the push chair for motor disabled children.

machined a rough shape model and an accurate shape model of the human face as shown in Figure 6.2. We have extracted zigzag tool paths straightforwardly from the numerical functions as described in Section 5.5.1. In the software version with which this comparison has been made, no selection of the best zigzag direction, as described in the previous chapter, had been implemented yet.

The most important aspect in the evaluation of the free space boundary following strategy and the slicing strategy is the comparison of the machining time. We have estimated the machining time with the model from Section 5.5.1. Also, we have computed the lower bound on the machining time for tool paths that are extracted from a single grid map. Figure 6.2 shows that the face model contains fragments with a constant height. Therefore, the lower bound on the machining time is not valid, because successive line segments connecting points with the same height are replaced by one line segment.

Table 6.1 gives an overview of some practical results. From the table we can make the following observations and conclusions:

- Although the length of the sliced paths is almost the length of the "free space boundary following" paths, machining with the sliced paths is almost four times faster, because the sliced paths consist of fewer line segments. Therefore, we decided to generate only sliced tool paths in the experiments described in Section 6.3.

Figure 6.2 The rough face model machined with a grid resolution of 2 mm (left) and a more accurate face model machined with a grid resolution of 1 mm (right).

- For a grid size larger than one millimetre machining takes much more time than computing. Hence, reducing the machining time is more important than reducing the computing time.
- For the push chair the lower bound on the machining time is not much lower than the estimated machining time. Therefore, the machining time can only be reduced significantly, if the zigzag machining pattern described in Section 5.5.1 is replaced by another machining pattern.
- Halving the grid resolution from 2.0 mm to 1.0 mm increases the computing time for the human face model by a factor 9.9 and for the push chair model by a factor 5.8. From a theoretical analysis it follows that the computing time of the tool path generation algorithm from Chapter 4 is dominated by the computation of the Minkowski sum of the numerical functions that represent the CAD model and the tool holder model. Computing this Minkowski sum with a straightforward implementation requires $O(mn)$, where m and n denote the number of grid elements of the CAD model and the tool holder model, respectively. Therefore, halving the grid resolution, increases the number of grid elements of both numerical functions

Shape	**Human face**	**Human face**	**Push chair**	**Push chair**
Size (mm)	185×206×116	185×206×116	510×948×269	510×948×269
Grid size (mm)	2	1	8	4
Milling tool	small	small	large	large
Computing time (min)	8	80	1	7
Required memory (Mb)	0.44	1.29	0.29	0.86
Free space boundary paths				
Path length (mm)	174146	324377	540213	874170
Number of line segments	74478	308352	72880	232633
Mean segment length (mm)	2.34	1.051	7.41	3.76
Estimated machining time (min)	253	954	307	855
Measured machining time (min)	-	-	-	-
Sliced paths				
Path length (mm)	156329	278340	421207	865716
Number of line segments	18552	69678	16279	60359
Mean segment length (mm)	8.43	3.99	25.87	14.33
Estimated machining time (min)	71	231	81	250
Measured machining time (min)	72	232	-	-
Lower bound machining time (min)	66	245	59	208

Table 6.1 Overview of practical results. The small milling tool is a ball-end tool with tool length 19 mm and tool radius 3 mm. The large milling tool is a ball-end tool with tool length 50 mm and tool radius 12 mm. The size of the tool holder is 238 mm × 156 mm × 248 mm.

---

with a factor 4 and increases the computing time of the Minkowski sum with a factor 16. Hence, for smaller grid sizes, it is also necessary to optimize the computing time, e.g. by implementing the Z-pyramid algorithm described in Chapter 4 or an algorithm based on contour processing (Young et al. 1981, van Vliet & Verwer 1988, Vincent 1991).

To study the volume removal rate against the time we plotted in figure 6.3 the volume removed as a function of time. The volume removal rate decreases with the time, because at the end of the milling process the shape boundary is followed. Since in general these paths contain of lot of short line segments, the milling process is slowed down. Therefore, following the shape boundary is the bottle-neck in the machining process.

6.3 The evaluation test of the MAOS method

6.3.1 Free-form objects used in the evaluation

We evaluated the computation of a limited set of tool access orientations as well as the path generation process with these tool access orientations. For this evaluation we used three free-form objects that define typical design engineering products with increasing complexity and with different morphological character:

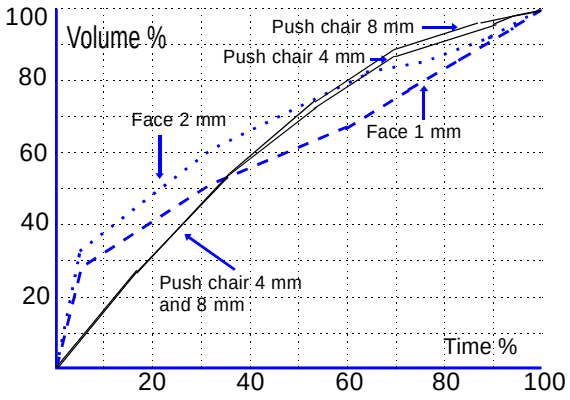
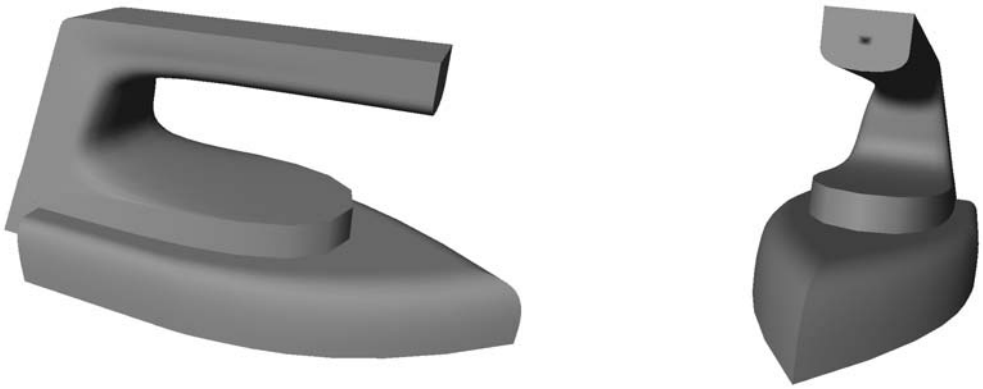


Figure 6.3 Graph of volume removal against time.



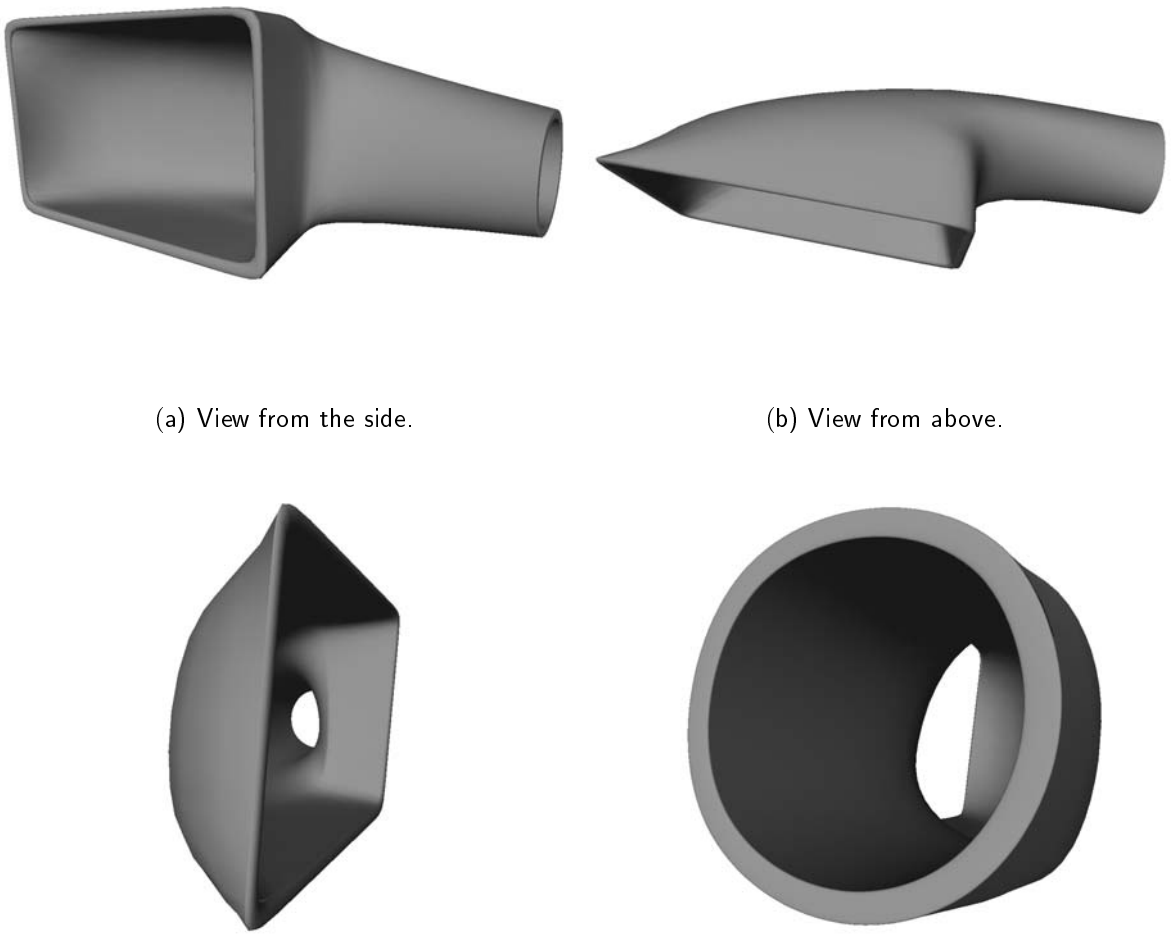
(a) The side of the iron object.

(b) The front of the iron object.

Figure 6.4 The iron object seen from two viewpoints.

- An iron (see Figure 6.4) with length 220 mm, width 100 mm and height 110 mm, that has been modelled by J.M. van der Made. Since the accessibility of the boundary of the iron shape is not limited severely, the complete boundary will be accessible with a few directions.

- A car mirror housing (see Figure 6.5) with length 240 mm, width 84 mm and height 100 mm, that has been modelled by S. Strassburger. The tunnel in the shape limits the accessibility of the boundary severely. Still, each part of the boundary will be accessible. Therefore, we expect that many directions will be needed to access the



(a) View from the side.

(b) View from above.

(c) Looking through the tunnel in the model.

(d) Looking at the other side of the tunnel.

Figure 6.5 The car mirror housing object seen from four viewpoints.

boundary completely.

- A pump (see Figure 6.6) with length 258 mm, width 77 mm and height 145 mm, that has been modelled by A. Valkenhoff and that has been improved further by L. Lennings. The pump contains a big internal cavity. Some part of the boundary inside the cavity is not accessible. The accessibility of other parts of the boundary is limited severely. Therefore, we expect that many directions will be needed to access the accessible part of the boundary.

(a) The side of the model. (b) The hollow pump has a big cavity.

Figure 6.6 The pump object seen from two viewpoints.

Because a new tool holder has been mounted on the sculpturing robot, the experiments in this section have been performed with another tool holder than the experiments in the previous section. Also an extra cylinder with length 190 mm and radius 100 mm is added to the tool holder object. This cylinder is not a part of the physical tool holder. Since this extra cylinder is modelled such that it always encloses the last two robot links, collision between these links and the stock-in-progress is avoided. The size of this new tool holder object is 390 mm $\times$ 222 mm $\times$ 230 mm. The milling tool is a ball-end tool with tool length 19 mm and tool radius 3 mm.

In the remainder of this section we show for each object the computed tool access directions, the computed tool access orientations, and the final result of the milling process. Also, we used the SRSIM software (Walstra & Vergeest 1994) to check the machining process as well to compute an accurate voxel model of the shape model.

6.3.2 Computation of the tool access directions

We used the algorithm from Section 3.4.4 to computed a limited set of tool access directions as follows. First, the accessibility for every pair of tool access directions is computed. The two tool access directions, with which the maximal number of voxels is accessible, are selected. Next tool access directions are selected one by one, until at least 95 % of the boundary voxels is accessible. Computing the tool access directions for the iron, the car mirror housing and the pump required 119, 165 and 135 minutes of computing time, respectively.

The tool access directions are shown in the Figures A.4 and A.5 in Appendix A in conjunction with a color coded voxel model. A voxel that has the same color as a tool

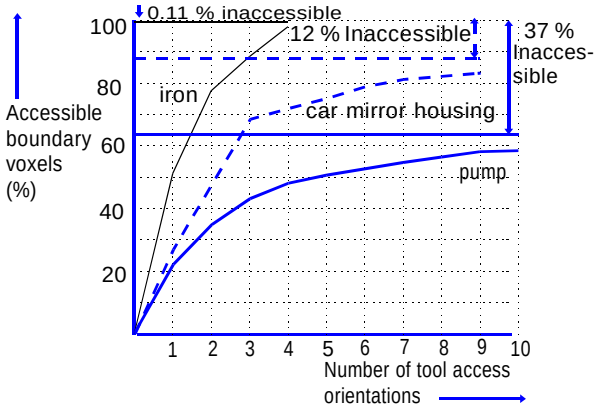


Figure 6.7 Percentage of accessible boundary voxels as function of the number of selected tool orientations.

access direction is accessible with that tool access direction. The blue voxels are accessible with the first tool access direction, the yellow voxels are accessible with the second tool access direction. Purple, light green, light blue, orange, pink, brown, light brown and light yellow voxels are accessible with the third, fourth, fifth, sixth, seventh, eight, ninth and tenth tool access direction, respectively. The boundary voxels for which no tool access orientations are computed, are colored dark green. Red voxels are not accessible with any tool access direction.

We have decided to use a 6-frequency icosahedron to select tool access directions. As described in Section 3.4.1 this icosahedron provides a set of 720 almost uniformly distributed normals on the unit sphere, from which the tool access directions are selected. We have set the voxel resolution to 6 mm.

Figure 6.7 shows, the percentage of accessible boundary voxels as a function of the number of selected tool access directions. Also, the percentage of the boundary voxels, which are not accessible by any tool access direction, is indicated.

Figure A.4 and Figure A.5 show that the selection of tool access directions takes into account the morphological character of the test objects. For the car mirror housing most of the directions access the channel and for the pump most of the orientations access its big internal cavity. The large percentage of inaccessible voxels for the pump and the car mirror housing are due to thin parts in these models. If a voxel contains two parts of the boundary of a thin part and the normals of these parts are approximately opposite to each other, then always an empty accessibility map will be computed and the voxel will be marked as inaccessible.

6.3.3 Computation of the tool access orientations

With the method from Section 5.4 the tool access directions have been extended to tool access orientations as shown by Figure A.6 in Appendix A. The best twist angle value

has been selected from 60 values uniformly distributed between 0 and 360 degrees. The iron and the car mirror housing have been computed with a support structure height from 300 mm. Since the direction selector selected two direction from below to machine the pump, the support structure height has been set to 500 mm to avoid collision between the tool holder and the turn table. Computing the tool access orientations for the iron, the car mirror housing and the pump required 67, 87 and 203 minutes of computing time, respectively.

6.3.4 Path generation

With the SR system we have machined a shape model of the iron. With the SRSIM simulator we have simulated the milling process for the car mirror housing and the pump. Also we have computed and displayed the car mirror housing and the pump shape models that can be fabricated with the SR system. We have used the model from Section 5.5.1 to estimate the machining time.

Figure A.7 in Appendix A shows the iron shape model. A comparison of this shape model with the geometry of the iron object from Figure 6.4 shows clearly that an amount of foam could not be removed. This foam is not accessible with the tool access directions from Figure A.4(a) due to collision avoidance between the stock-in-progress and the tool holder. The upper photo shows one side of the iron, which has been machined accurately with the blue direction from Figure A.4. The lower photo shows the opposite side, which has been machined more roughly with the yellow and green directions from Figure A.4. The side of the iron machined with the blue direction is almost perpendicular to that direction. The side of the iron machined with the green and yellow directions are almost parallel to these directions. The big difference in the surface quality at different sides of the iron model is explained as follows. For the zigzag tool path pattern the grid size d denotes the distance between the projections of two adjacent tool paths in a plane perpendicular to the tool access orientation. The distance of these tool paths on a surface is equal to $d/cos(\alpha)$, where α denotes the angle between the surface normal and the tool access direction. Unwanted ridges will be higher if the distance between the tool paths is larger. Therefore, machining with tool access directions, that are almost parallel to the machined surface, provides a poor surface quality.

Figure A.8 in Appendix A shows shape model of the car mirror housing. Also, Figure A.9 in Appendix A the pump shape model that has been computed by SRSIM. A comparison of the shape model with the car mirror housing object from Figure A.8 shows that almost all foam could be removed.

The simulation of the machining process of the pump revealed a shortcoming in the implementation of the MAOS path planner. Figure 6.8 shows that for a tool access direction from below the tool holder can still collide with the turn table. In the implementation of the path planner and the orientation selector it has been assumed that the tool can be positioned always such that the tool tip touches a face of the box enclosing the shape model. But the path planner generates the path in a box, that encloses the shape model as well as the support structure. For a machining direction from below the path planner

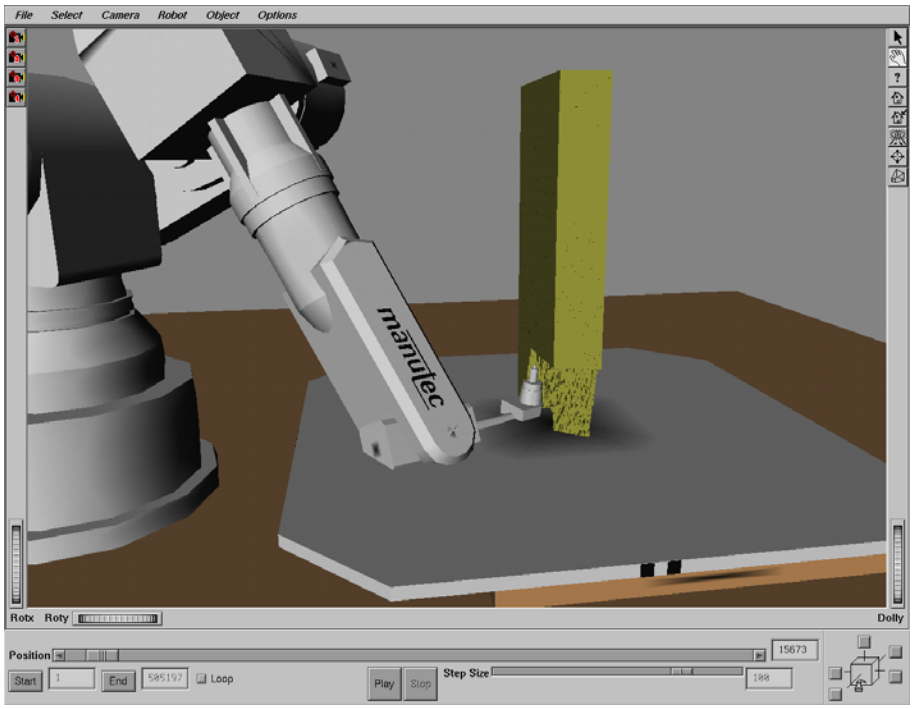


Figure 6.8 SRSIM shows a collision between the tool holder and the turn table.

positions the tool tip erroneously at a face of this box. If we would use a box that encloses the shape model only, then we may introduce collisions between the tool holder and the support structure. Therefore, solving this shortcoming is part of the ongoing research that will be discussed in Section 6.4.

Figure 6.9 shows the pump shape model that has been computed by SRSIM. Because of the collision between the tool holder and the turn table this model cannot be fabricated with the SR system.

Table 6.2 provides an overview of the practical results. All machining times with exception of "the measured machining time" have been estimated with the model described in Chapter 5. "CPU time for stock update" denotes the time that is required to update the grid maps representing the stock-in-progress as described in Section 5.5.2. The "Machining time after first round" denotes the total machining time after for each tool access orientation milling paths have been generated exactly once by the round-robin Algorithm 5.1. The "Worst case machining time" denotes the time that is required for machining if the path planner always takes the worst option to generate zigzag paths as described in Section 5.5.1. Because, the stock-in-progress is shrunk, it may be possible to remove some more material by repeating the milling process. Therefore, Algorithm 5.1 repeats the milling process and generates milling paths for only those tool access orientations with which material is removed. The lower bound on the machining time is the sum of the lower bounds for the machining time for each tool orientation from the first

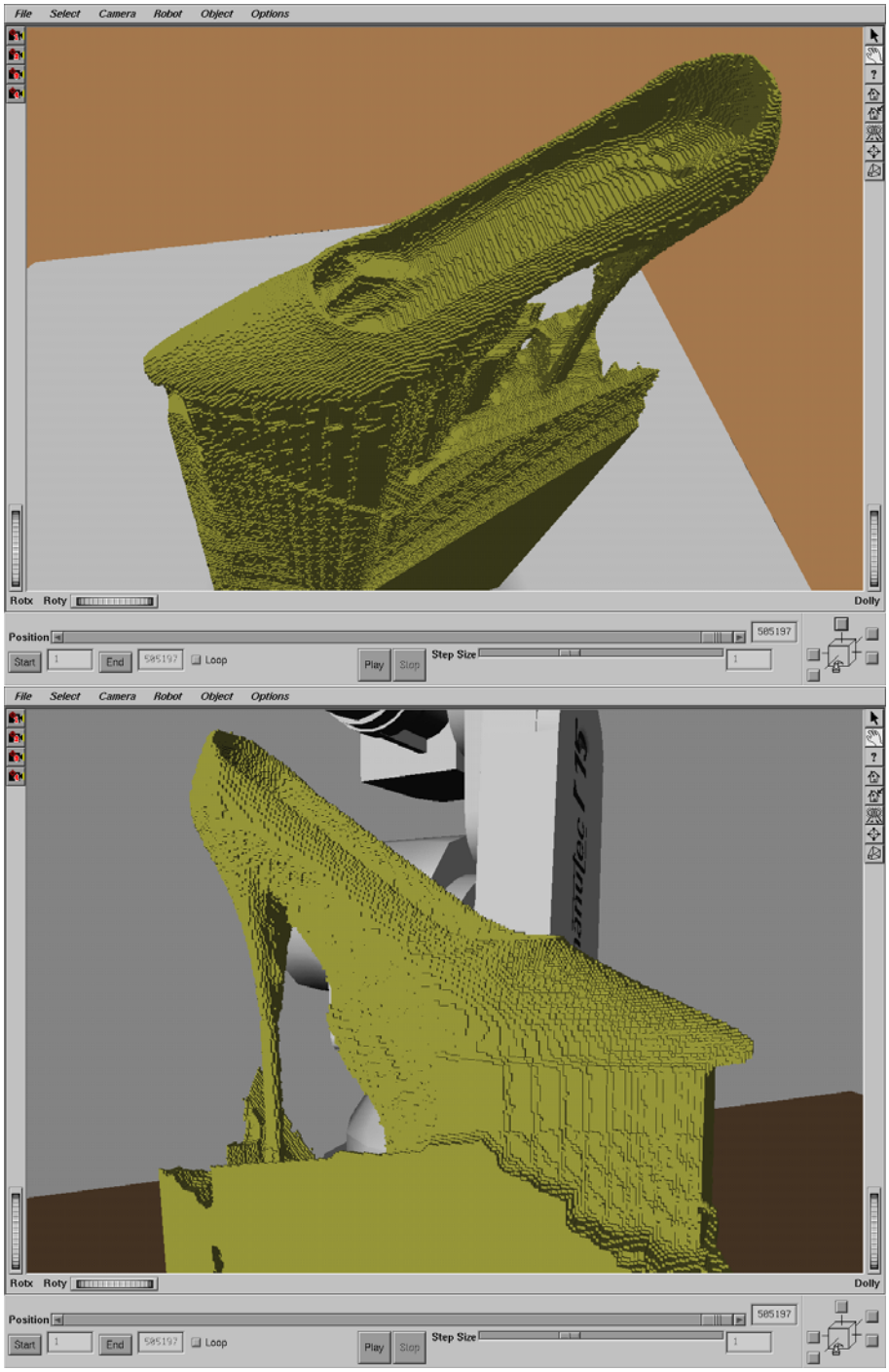


Figure 6.9 SRSIM displays with a 1 mm voxel resolution the pump shape model.

Shape	Iron	Car mirror housing	Pump
Model size (mm)	220×100×110	240×84×100	258×77×145
Height of support structure (mm)	300	300	500
Number of tool access directions	4	9	10
CPU time for direction selector (min)	119	165	135
CPU time for orientation selector (min)	67	87	203
CPU time for path planning (min)	251	1248	3451
CPU time for stock update (%)	32	87	76
Total CPU time	437	1500	3789
Lower bound machining time (min)	201	587	923
Worst case machining time (min)	603	2117	2980
Total machining time (min)	489	1568	2559
Measured machining time (min)	436	-	-
Material removed (mm^3)	$3.59*10^6$	$5.48*10^6$	$9.57*10^6$
Machining time after first round (min)	326	876	1464
Material removed after first round (%)	99.7	98.7	98.8

Table 6.2 Overview of practical results. All models have been obtained with a grid resolution of 2 mm.

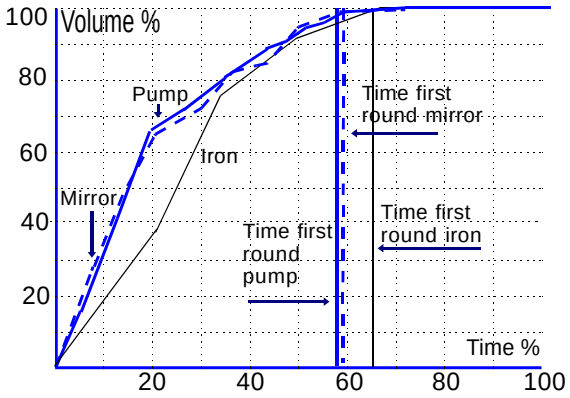


Figure 6.10 volume removal against time for the iron (thin line), the car mirror housing (dashed line) and the pump (solid line). For the iron, the car mirror housing and the pump 99.7 %, 98.7 % and 98.8 % of the material, respectively was removed after the first round of the milling process.

round.

From Table 6.2 and Figure 6.10 we can draw the following observations and conclusions:

- Table 6.2 shows that in the first round almost all material is removed in about 60 % of the machining time. Since the result after the first round is almost the same as the final result, a lot of machining time as well as computing time can be saved by

machining only the first round.

- The lower bound on the machining time is about two third the machining time of the parts from the first round. We conclude that with our zigzag tool path pattern it is not possible to machine much faster.
- Compared to the "worst case" the selection of the best option to generate zigzag paths saves about 20 % machining time.
- The bottleneck in the computation of the milling process is the time needed to update the grid maps representing the stock-in-progress.
- A shape model of a simple model like the iron can be fabricated in one day. Due to the high computation time and the machining time it takes a few days to obtain a complex shape model like the car mirror housing model or the pump model.

6.4 Ongoing research

Especially for machining models from below and machining large model, it is mandatory to detect and avoid collisions between the robot links and the various obstacles, including the shrinking stock-in-progress. The example of the pump showed that for a tool access orientation from below it is not sufficient to represent the free space boundary by a single grid map. One grid map is needed to avoid collision between the tool holder and the stock-in-progress and one to avoid collision between the tool holder and the turn table.

(Kovács 1998) and (Kuczogi 1998) have recently developed a method to avoid collision between (a) the robot links and the shrinking stock-in-progress, (b) the robot links and the obstacles including the turn table and (c) the tool holder and the obstacles including the turn table. Given a tool access orientation they enclose, the free space by an upper and a lower grid map. To implement collision avoidance efficiently, they approximate the tool holder, the stock-in-progress, the turn table, and each of the robot links, by a collection of spheres as shown in Figure 6.11. Given a tool access orientation $\mathbf{o}$, their method shrinks a box, that encloses the initial object and the support structure, to the free space. The upper and lower grid map represent two faces of this box that are perpendicular to $\mathbf{o}$. A large number of points are sampled in this box and it is tested whether the tool tip can be positioned with orientation $\mathbf{o}$ without any collision. The results of these tests are used to adapt the lower and upper grid map, such that these maps enclose the free space. It is planned to integrate this method into the MAOS software.

6.5 Recommended extensions to MAOS

Based on the evaluation we recommend the following extensions to MAOS:

The shape model of the iron shows, that machining with directions that are almost parallel to the machined surface, produces unwanted ridges. Such tool access directions can be selected, because the local visibility map at a point $\mathbf{p}$ with normal vector $\mathbf{n_p}$

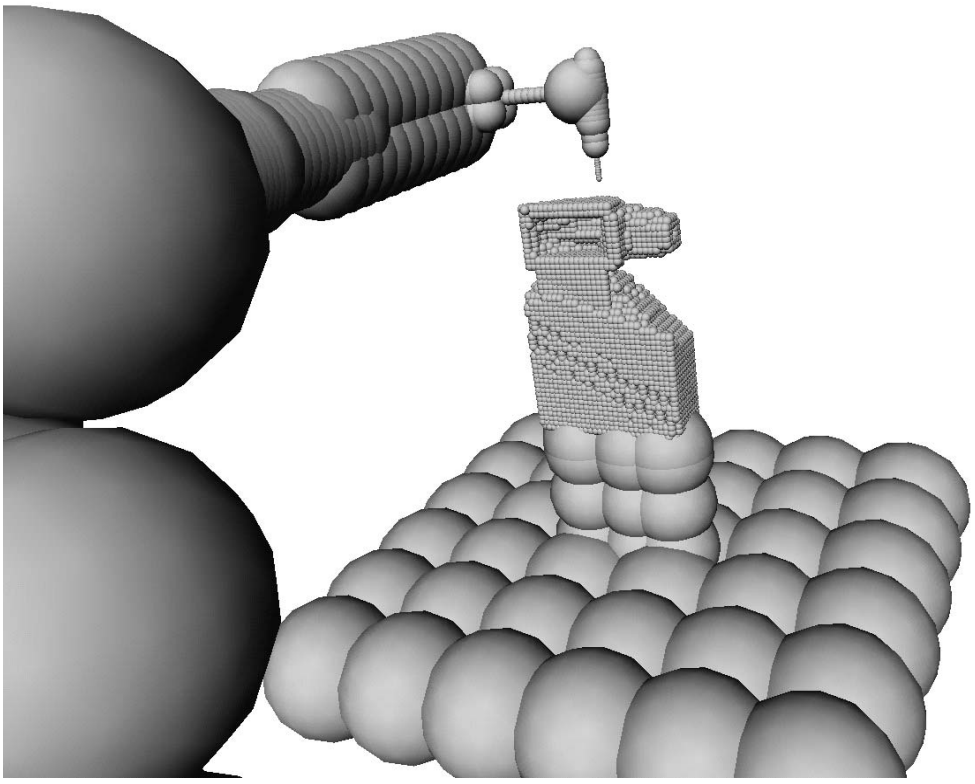


Figure 6.11 Spherical approximation of the stock-in-progress, the robot, the tool holder and the turn table.

contains directions from a hemisphere, bounded by the tangent plane at $\mathbf{p}$. Therefore the selection of tool access directions should be limited to a visibility cone. Such a cone should contain only vectors $\mathbf{v}$, for which the angle between $\mathbf{v}$ and $\mathbf{n_p}$ is smaller than e.g., 15 degrees.

In the example of the iron, more material would be removed with other tool access directions than with the selected tool access directions. Our method did not select good tool access orientations, because it performs only accessibility analysis for the milling tool. A method, that performs accessibility analysis for the tool holder will select tool orientations with which more material can be removed.

In the current implementation, MAOS finishes the machining process if no more material can be removed. Since about 99 % of the material is removed in about 60 % of the machining time, a lot of computation and machining time can be saved by finishing the machining process earlier. Instead of repeating the machining process for the same tool access orientations, it would be more effective to add a few new tool access orientations.

The machining time can be further reduced if paths that do not remove material are avoided as much as possible. To support this the Minkowski algorithm should divide the free space into a machining space in which material is removed and a "out of material

space".

The lower bound on the machining time shows that a significant reduction of the machining time cannot be obtained with the zigzag pattern from Section 5.5.1. Therefore, methods to smooth machining paths should be investigated. A smoothed path can be followed at much higher speed, but collision avoidance is more difficult. Moreover, a smoothed path may not remove the same material as the original path would. If a big and a small milling tool is available a roughing stage by the big milling tool using the Minkowski method and a finishing stage by the small milling tool with smoothed tool paths may save a lot machining time

The bottleneck in the computation of the milling process is the time needed to update the grid maps representing the top of the stock-in-progress for each tool access orientation. In the MAOS implementation, after the modification of one of these grid maps, it is necessary to update all other grid maps. A representation of the stock-in-progress independent on the selected tool access orientations, like a spherical approximation, would not require this computational expensive update procedure. To apply the Minkowski based machining method, it would be necessary to develop also conversion methods between grid representations and spherical representations. In combination with this spherical representation, collision detection methods based on a hierarchy of spheres enclosing the volume swept by the robot links and the tool holder (Verwer 1990) can be applied.

7

Conclusions

With the new generation of 3D CAD systems the designer can quickly create a number of free-form objects and use virtual modeling to make a first comparison. Both virtual and physical shape modeling techniques support conceptual shape design, but physical shape models provide feedback to the designer going beyond visual presentation. For physical shape modeling, NC milling is the main competitor to systems based on incremental methods. Completely automated NC milling would have a significant impact on fast physical shape modeling. The implementation of 5-axis or 6-axis milling methods extends the class of mid-size and large complex shape models that can be fabricated automatically. Compared to 5-axis milling 6-axis milling offers the following two benefits: (a) with help of a sixth rotational axis often interference between the tool holder and the stock-in-progress can be avoided, and (b) with 6-axis milling only one set-up is needed to machine a shape model. Therefore, in Chapter 1 we stated the main research question in this thesis: how can we support physical shape modeling by 6-axis milling processes?

Our research aimed at solutions for the issues given below:

- **Interference avoidance between stock-in-progress and the tool holder**
  The tool holder should never interfere with the stock-in-progress.

- **Interference avoidance between the shape model and the milling tool**
  The milling tool should never interfere with the shape model.

- **Extension of the shape domain of manufacturable objects**
  Advanced machining algorithms should make full use of the capabilities of 6-axis milling.

- **Fabrication of shape models of initial free-form objects**
  In conceptual design the product geometry may be only roughly specified. Therefore, the software has to be able to process initial free-form objects, that are roughly specified.

- **Machining large models**
  To obtain a large model within a few days, both the computation of the machining process and the machining process itself should be fast.

In Chapter 2 we have investigated the problem of sculpturing shape models taking into account six degrees of freedom: three degrees to position the milling tool and three to orient the milling tool. To reduce the complexity we decomposed this problem into a *tool accessibility problem* and a *tool path generation problem*. In the case of the tool accessibility problem a limited number of tool access orientations are extracted from the geometry of a free-form shape. The tool path generation problem is simplified, because for each of these tool access orientations tool paths are generated separately. With these tool paths as much material as possible is removed, while avoiding interference between the tool and the shape model, as well as between the tool holder and the stock-in-progress.

In Chapter 3 we have described a new approach to the tool accessibility problem. Given the shape F of an initial free-form object, represented by a set of B-spline surfaces, our method determines a limited number of tool access orientations. We have developed and implemented a new method that transforms F's shape into a voxel representation. Each voxel is classified either as an exterior, boundary or interior voxel. For each voxel containing part of F's boundary, an accessibility map is computed, recording a finite number of tool access directions. From these accessibility maps a limited number of tool access directions are selected and extended to tool access orientations. In the literature we found only references to methods, that store accessibility maps per surface. These accessibility maps contain only directions with which the complete surface is accessible. To access surfaces like spheres and cylinders completely, more than one tool direction is needed. So, for such a surface an empty accessibility map will be computed. Therefore, we developed a method, that computes accessibility maps for each boundary voxel. Since this requires the computation of many accessibility maps, an efficient procedure to compute these maps was needed. Hence, we developed and implemented an approximate method to compute accessibility maps. The evaluation presented in Chapter 5 showed that the computed tool access orientations take into account the morphological character of the initial free-form object.

In Chapter 4 we have shown that Minkowski operations and spatial planning provide powerful tools to describe tool path generation, taking into account interference avoidance between the tool and the shape model, and between the tool holder and the stock-in-progress. We have described and compared freeform shape machining algorithms that use fixed tool orientations. We have decomposed the interference avoidance problem into a tool holder spatial problem and a tool spatial problem. In spatial planning an object is placed or moved among one or more obstacles without interfering them. In the case of the tool holder spatial planning problem, the tool holder is the object and the stock-in-progress is the single obstacle. In the case of the tool spatial planning problem, the tool is the object and the shape to be obtained is the single obstacle. In both cases the tool holder and the tool are shrunk to the same point $\mathbf{o}$. We have described the forbidden region for positioning $\mathbf{o}$ as the union of the dilated stock and the dilated model. Finally, we have described a "free space boundary following" machining strategy and a

slicing machining strategy that both extract tool paths from the boundary between the forbidden region and the free space. Although tool path generation with a fixed tool orientation is very well-know in the literature, this formal approach, including tool and tool holder interference avoidance, has not been described before.

In Chapter 5 we have described the multiple access orientation sculptor (MAOS) software that implements our method for the Sculpturing Robot system at Delft University of Technology.

In Chapter 6 we have evaluated the MAOS software package for three typical design engineering products with a different morphological character.

In summary, our contributions to the research issues stated at the beginning of this chapter are:

- The MAOS path planner avoids interference between the tool and the shape model, and between the tool holder and the stock-in-progress. Due to the limited scale implementation of the path planner, collision both between the tool holder and the turn table and between the robot links and the stock-in-progress is not avoided. Solving this issue is a part of the ongoing research that has been described in the previous chapter.

- MAOS extends the shape domain of manufacturable objects significantly. Free-form objects like the iron and the car mirror housing from Section 6.3.1 can be machined properly, because the complete boundary of these models is accessible with a limited number of tool orientations. A shape like the pump from Section 6.3.1 cannot be machined completely, because it contains an internal cavity.

- The examples from Section 6.1 show that MAOS can process initial free-form objects that are roughly specified. MAOS checks whether free-form objects properly represent a shape within a user-defined accuracy. For the two presented examples MAOS detected the errors in their shape representation properly. After a simple correction of these errors MAOS accepted these free-form objects.

- Due to the high computation time and the high machining time, it takes about two days to machine a medium sized shape model like the car mirror housing with length 240 mm, width 84 mm and height 100 mm. To obtain medium sized models faster and to obtain large models within a few days both the machining time and the computation time should be reduced further as described in Section 6.5.

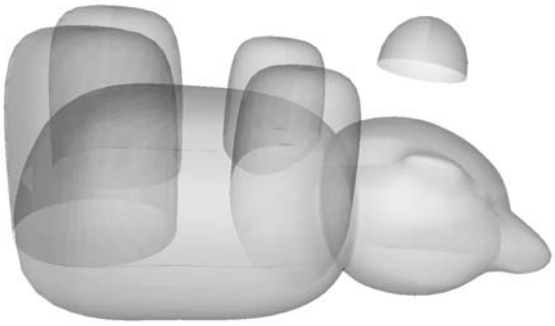
The research described in this thesis has led to (a) a new method to compute a limited number of tool access orientations by accessibility analysis of a free-form object, (b) a new formal method to describe tool path generation taking into account interference avoidance for both the tool and the tool holder, and (c) the development of a new approach to 6-axis milling of shape models of free-form objects with a sculpturing robot. This new approach has been successfully implemented and applied to machine medium sized shape models with the sculpturing robot.

The most serious restrictions of the present SR system are (a) the inherent partial inaccessibility of free-form objects that have a very complex shape or contain inner structures, and (b) the limitations on the size of the shape models that can be machined. To cope with this, it will be necessary to automatically decompose the free-form object into fully accessible portions, which can hence be machined and finally recomposed. One current approach to this so-called hybrid rapid prototyping is the fast fabrication of thick slices using flexible blade cutting (Horváth, Vergeest, Broek & de Smit 1998).

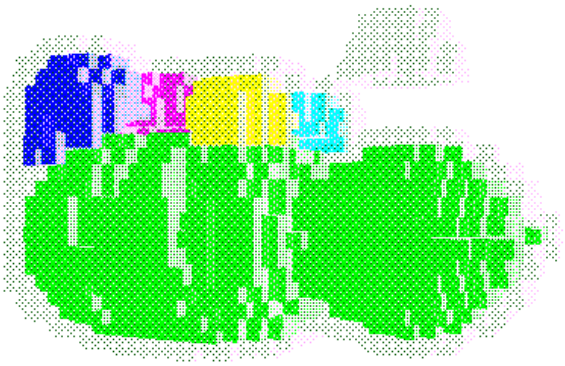
Appendix A

Color Figures

This appendix contains the color figures belonging to the evaluation described in Chapter 6.

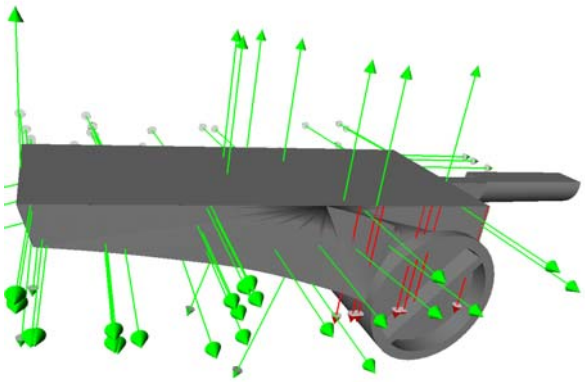


(a) Free-form object representing a bear. The nose of the bear is separated from its head by a gap broader then the user-selected geometric accuracy.

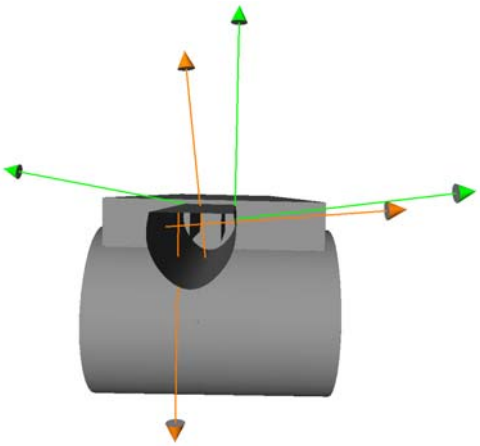


(b) The voxel representation of the bear. The interior volume of the bear is represented by 5 separate face-components colored green, blue, pink, yellow and light blue. The boundary volume of the bear is represented by 2 separate grey edge-components.

Figure A.1 Example of an erroneous free-form object consisting of two separate parts.

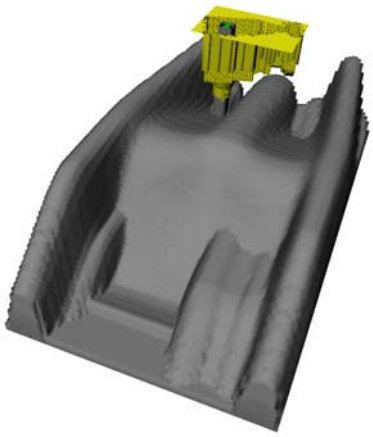


(a) The green and the red normals point in opposite directions.

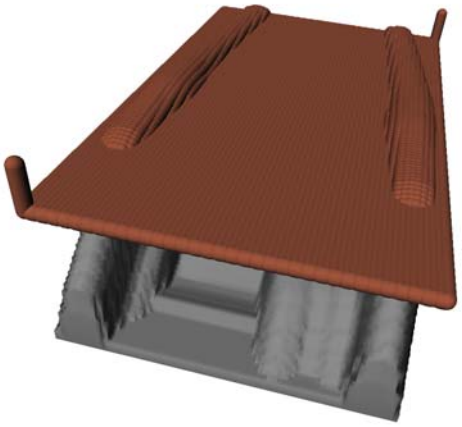


(b) The green and the orange normals point outwards at both surface sides.

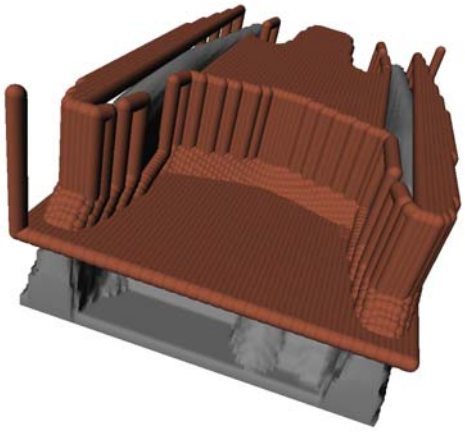
Figure A.2 Example of an erroneous free-form object with normals pointing outwards at both surface sides.



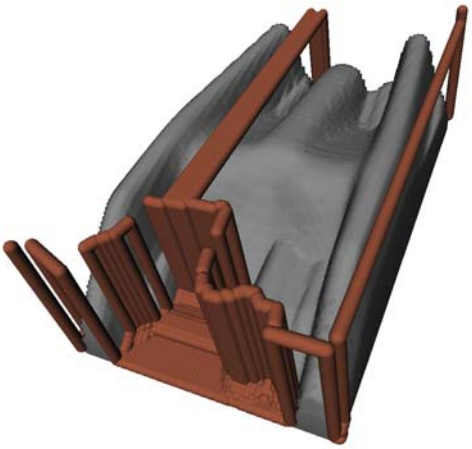
(a) Shape model of the seat of the push chair shown with tool holder model.



(b) Paths in stage 1.

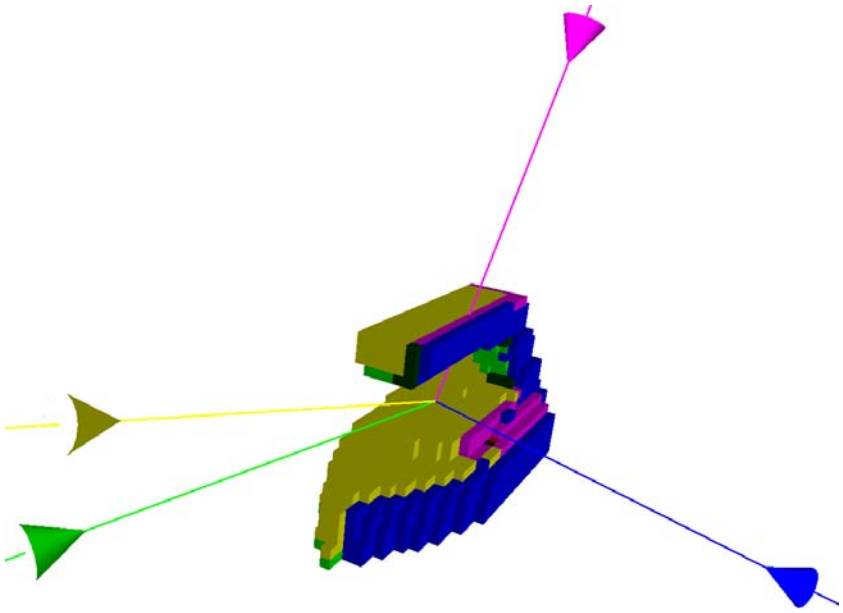


(c) Paths in stage 4.

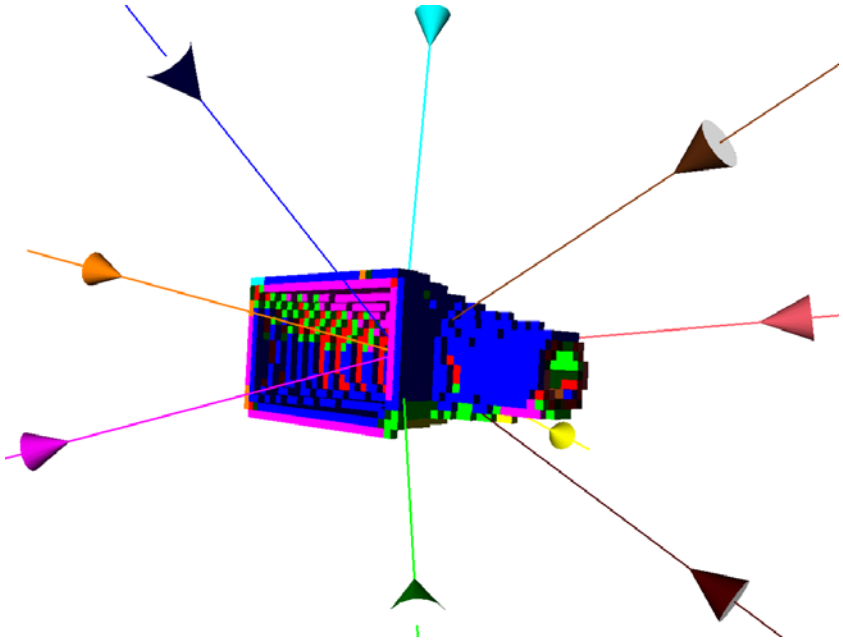


(d) Paths in stage 7.

Figure A.3 Simulation of the machining process of the shape model of the push chair.

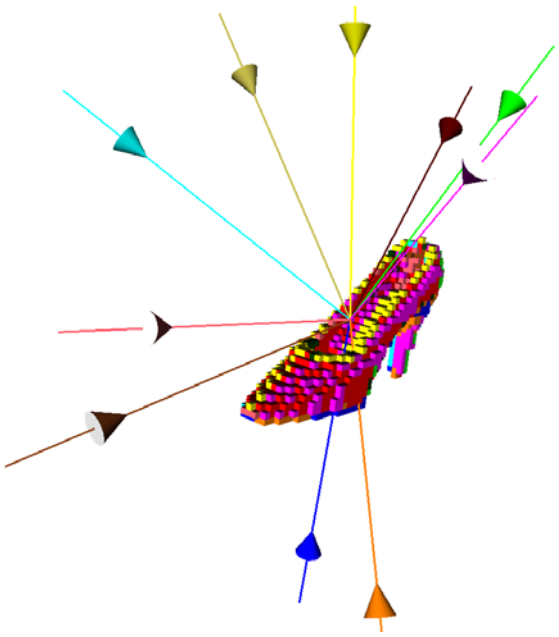


(a) The four tool access directions for the iron.

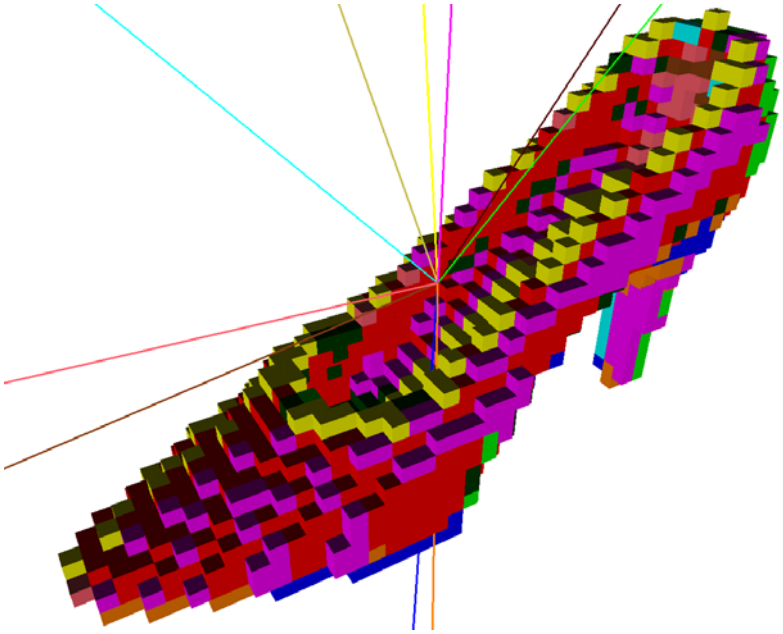


(b) The nine tool access directions for the car mirror housing.

Figure A.4 Selected tool access directions for the iron and the car mirror housing with which at least 95 % of the boundary voxels is accessible.

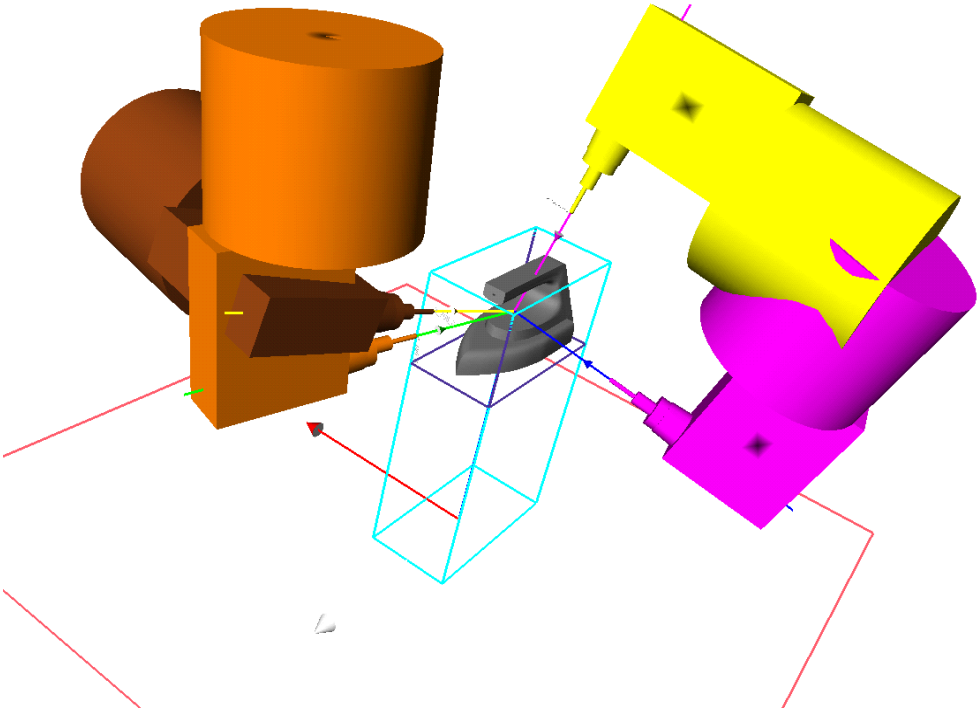


(a) The ten tool access orientations for the pump.

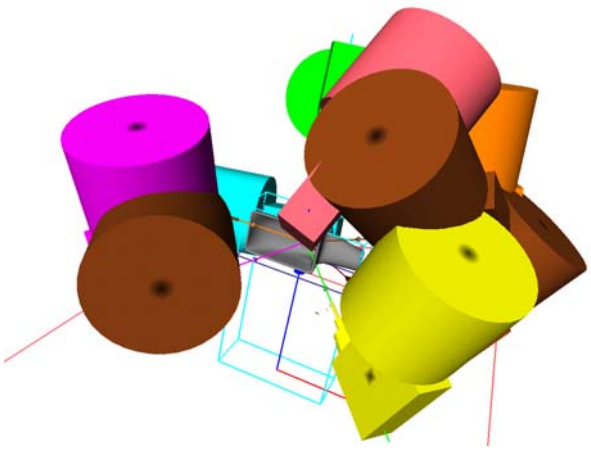


(b) Color coded voxel model of the pump. The color of a voxel is the same as the color of the direction with which it is accessible.

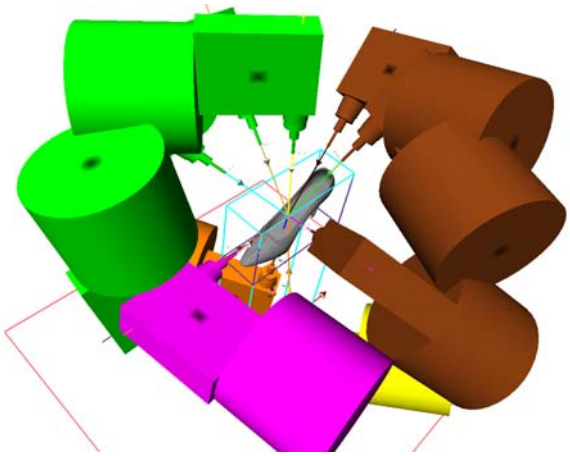
Figure A.5 Selected tool access directions for the pump with which at least 95 % of the boundary voxels is accessible.



(a) The four tool access orientations for the iron.



(b) The nine tool access orientations for the car mirror housing.



(c) The ten tool access orientations for the pump.

Figure A.6 Selected tool access orientations. For each tool access orientation the color of the tool holder denotes one of the eight possible turn table angles.

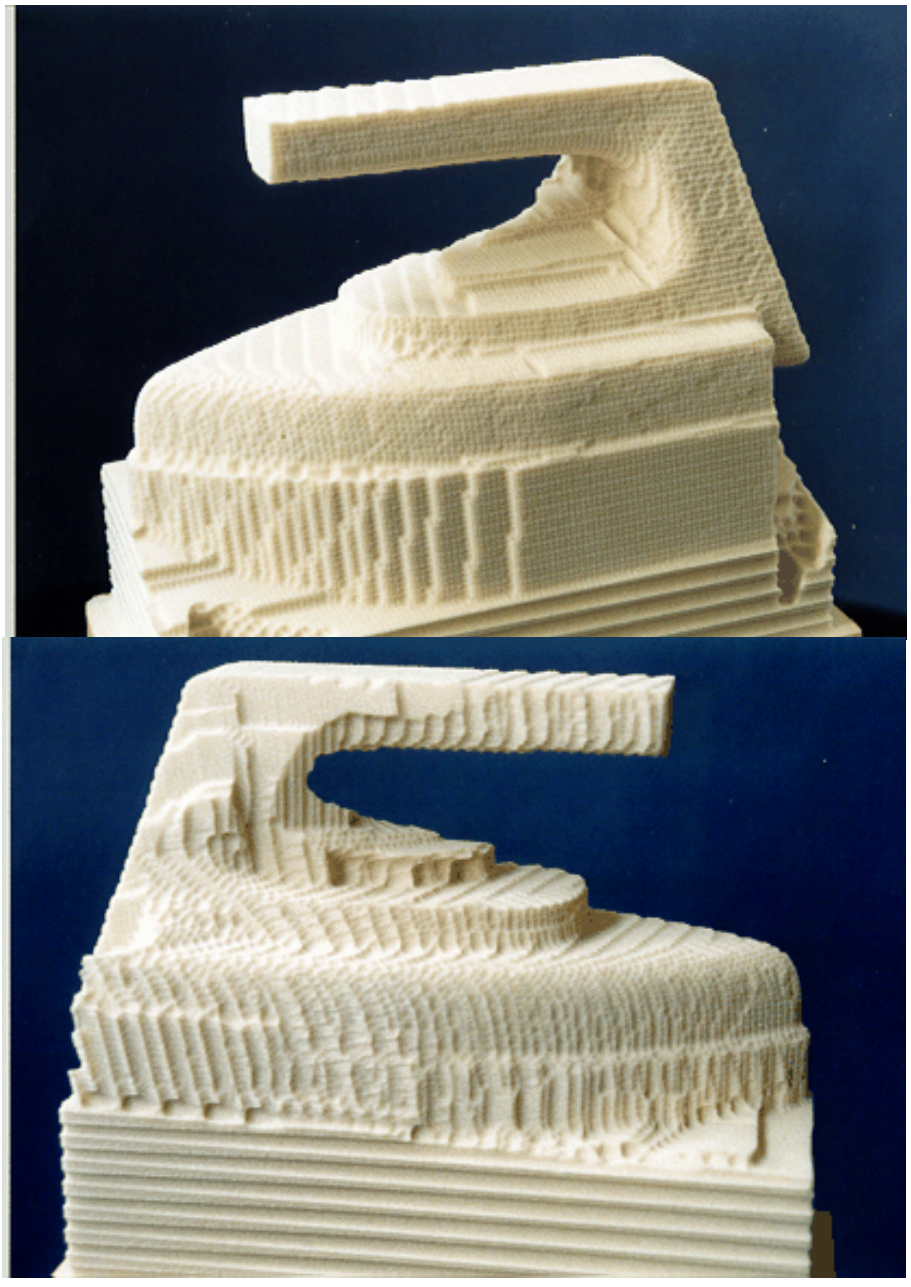


Figure A.7 The iron shape model.

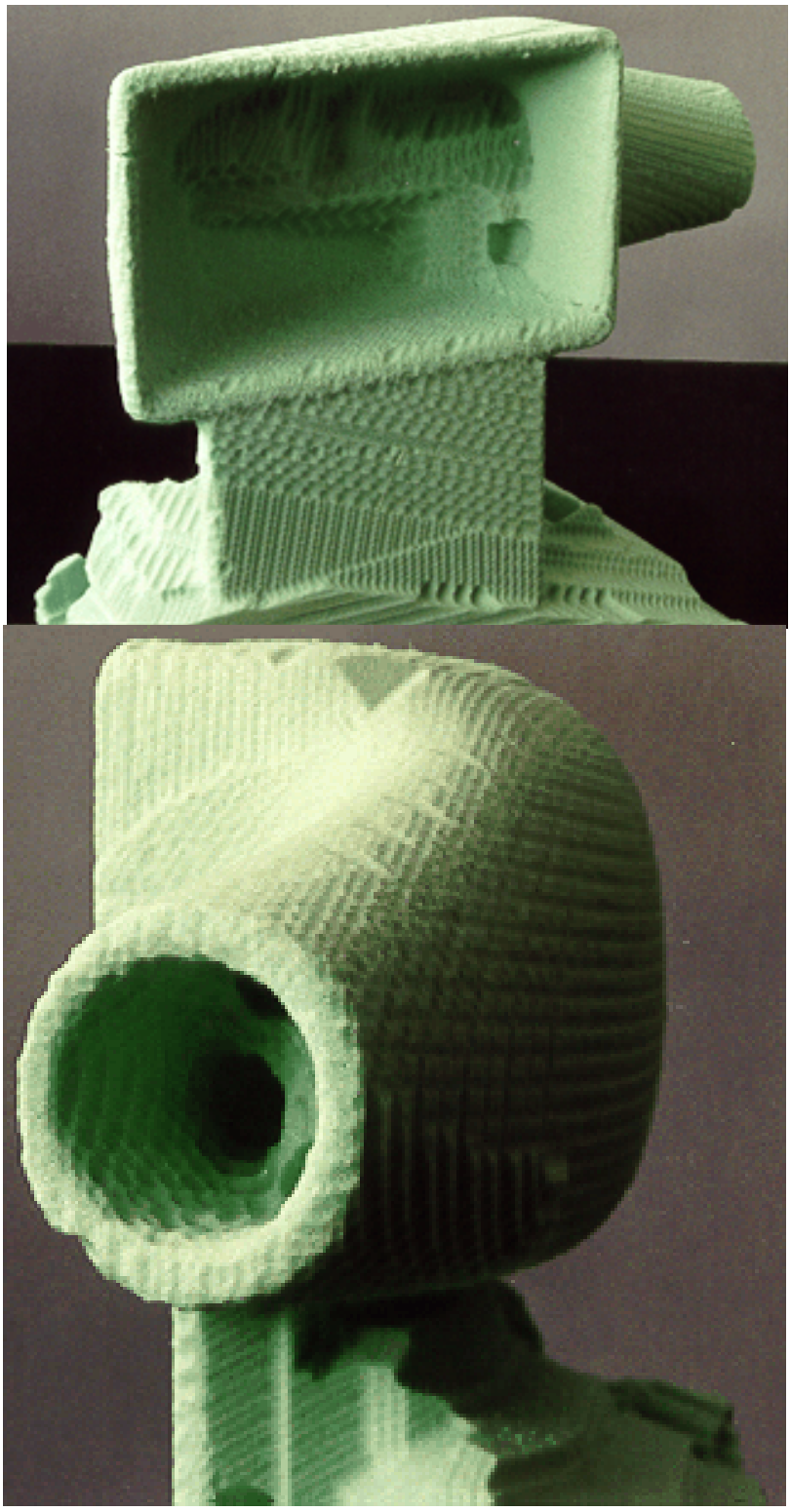


Figure A.8 The shape model of the car mirror housing.

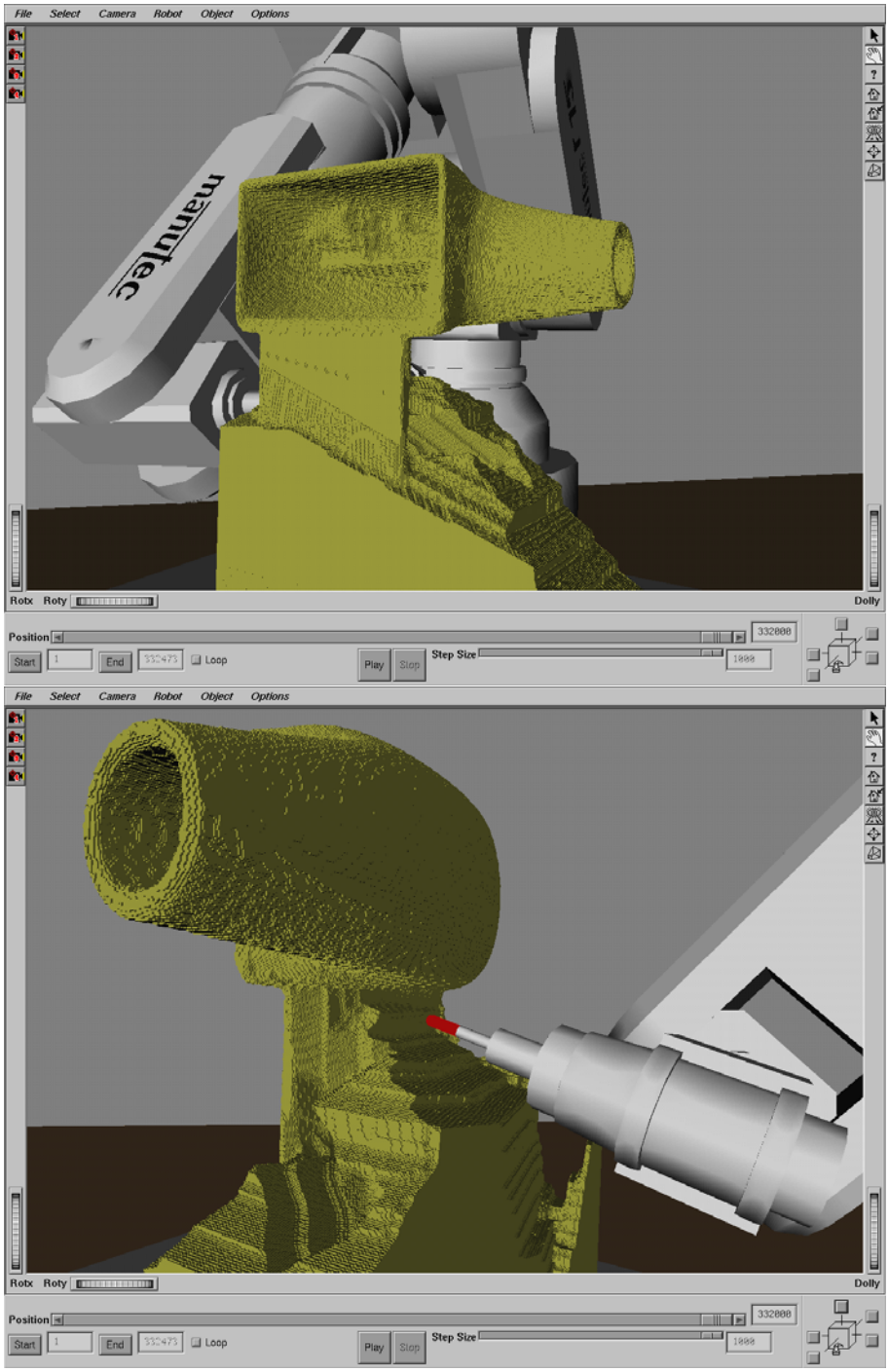


Figure A.9 SRSIM displays with a 0.5 mm voxel resolution the car mirror housing shape model.

Bibliography

Baxter, M. (1995), *Product Design*, Chapmann & Hall, London, UK.

Beaman, J., Barlow, J., Bourell, D., Crawford, R., Marcus, H. & McAlea, K. (1997), *Solid Freeform Fabrication: A new Direction in Manufacturing*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Boehm, W. (1996), Differential geometry II, *in* G. Farin, ed., 'Curves and Surfaces for Computer Aided Geometric Design: a Practical Guide, 4th. ed.', Academic Press, San Diego, CA, chapter 22, pp. 348–362.

Broek, J. & Vergeest, J. (1996), Calibration of a sculpturing robot system, *in* 'The Fourth International Conference on Control, Automation, Robotics and Vision (ICARV'96)', Nanyang Technological University, Singapore, pp. 2152–2156.

Brunet, P., Navazo, I. & Vinacua, A. (1993), 'A modelling scheme for the approximative representation of closed surfaces', *Computing Suppl.* **8**, 75–90.

Chen, C. & Kak, A. (1989), 'A robot vision system for recognizing 3-D objects in low-order polynomial time', *IEEE Transactions on Systems, Man, and Cybernetics* **13**(1), 1535–1563.

Chen, L., Chou, S. & Woo, T. (1993), 'Separating and intersecting spherical polygons: computing machinability on three-, four-, and five-axis numerically controlled machines', *ACM Transactions on Graphics* **12**(4), 305–326.

Cho, I., Lee, K. & Kim, J. (1997), 'Generation of collision-free cutter location data in five-axis milling using the potential energy method', *The International Journal of Advanced Manufacturing Technology* **13**, 523–529.

Choi, B., Chung, Y.-C. & Park, J.-W. (1995), Application and extension of Z-map model, *in* S. Shin & T. Kunii, eds, 'Computer graphics and applications; proc. of the Third Pacific Conference on Computer Graphics and Applications', World Scientific, Singapore, pp. 363–382.

Choi, B., Kim, D. & Jerard, B. (1997), 'C-space approach to tool-path generation for die and mould machining', *Computer-Aided Design* **29**(9), 657–669.

Choi, B., Park, J. & Jun, C. (1993), 'Cutter-location data optimization in 5-axis surface machining', *Computer-Aided Design* **25**(6), 377–386.

Chu, C.-C., Dani, T. & Gadh, R. (1997), 'Multi-sensory user interface for a virtual-reality-based computer-aided design system', *Computer-Aided Design* **29**(10), 709–725.

Craig, J. (1989), *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, Reading, Massachusetts.

Cross, N. (1994), *Engineering Design Methods : Strategies for Product Design*, second edn, John Wiley & Sons Ltd., Chichester, UK.

Dani, T. & Gadh, R. (1997), 'Creation of concept shape designs via a virtual reality interface', *Computer-Aided Design* **29**(8), 555–563.

de Berg, M., van Kreveld, M., Overmars, M. & Schwarzkopf, O. (1997), *Computational Geometry : Algorithms and applications*, Springer-Verlag, Berlin, Germany.

de Jager, P. (1998), Development of a New Slicing Methodology to Improve Layered Manufacturing, PhD thesis, Delft University of Technology.

Dijkstra, E. (1959), 'A note on two problems in connexion with graphs', *Numer. Mathem.* **1**, 269–271.

Dragomatz, D. & Mann, S. (1997), 'A classified bibliography of literature on NC milling path generation', *Computer-Aided Design* **29**(3), 239–247.

Edelsbrunner, H. & Mücke, E. (1994), 'Three-dimensional alpha shapes', *ACM Transactions on Graphics* **13**(1), 43–72.

Eggli, L., Hsu, C.-Y., Brüderlin, B. & Elber, G. (1997), 'Inferring 3D models from freehand sketches and constraints', *Computer-Aided Design* **29**(2), 101–112.

Elber, G. (1994), 'Accessibility in 5-axis milling environment', *Computer-Aided Design* **26**(11), 796–802.

Elber, G. (1995), 'Freeform surface region optimization for 3-axis and 5-axis milling', *Computer-Aided Design* **27**(6), 465–470.

Elber, G. & Cohen, E. (1995), Arbitrarily precise computation of Gauss maps and visibility sets for freeform surfaces, *in* 'Proc. Solid Modeling '95', Salt Lake City, Utah, USA, pp. 271–279.

Elber, G. & Cohen, E. (1996), Gouge detection and prevention of multi axis machining tools for freeform surfaces, *in* J. McCarthy, ed., 'Proc. of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference', ASME, New York. CD-ROM.

Farin, G. (1996), *Curves and Surfaces for Computer Aided Geometric Design: a practical guide.*, fourth edn, Academic Press, San Diego, CA.

Gan, J. (1992), 'Set-up orientations of workpieces for machining by three-axis numerical control machines', *Journal of Design and Manufacturing* **2**, 59–69.

Ghosh, P. (1990), 'A solution of polygon containment, spatial planning, and other related problems using Minkowski operations', *Computer Vision, Graphics and Image Processing* **49**, 1–35.

Giardina, R. & Dougherty, R. (1988), *Morphological Methods in Image and Signal Processing*, Prentice Hall.

Goldschmidt, G. (1991), 'The dialectics of sketching', *Creativity Research Journal* **4**(2), 123–143.

Gupta, P., Janardan, R., Majhi, J. & Woo, T. (1996), 'Efficient geometric algorithms for workpiece orientation in 4- and 5-axis NC machining', *Computer-Aided Design* **28**(8), 577–587.

Head, M., Kedem, G. & Prisant, M. (1994), Application of the ray-representation and a massively parallel special purpose computer to problems of protein structure and function: I. Methodology for calculation of molecular contact surface, volume, and internal free space, Technical report, Departments of Chemistry and Computer Science, Duke University, Durham, North Carolina, http://www.mbp.duke.edu/mbp/geom/method/master/master.html.

Horváth, I., Vergeest, J., Broek, J. & de Smit, A. (1998), 'Tool profile and tool path calculation for free-form thick-layered fabrication'. In review.

Hummels, C. & Stappers, P. (1998), Meaningful gestures for human computer interaction: Beyond hand postures, *in* 'The proceedings for the Third IEEE International Conference on Automatic Face & Gesture Recognition (FG'98)', IEEE Computer Society Press, Los Alamitos.

Hummels, C., Smets, G. & Overbeeke, K. (1998), An intuitive two-handed gestural interface for computer supported product design, *in* I. Wachsmuth & M. Fröhlich, eds, 'Gesture and sign language in human computer interaction: proceedings of Bielefeld Gesture Workshop 1997', Springer Verlag, Berlin.

Janardan, R. & Woo, T. (1997), Design and manufacturing, *in* J. Goodman & J. O'Rourke, eds, 'Handbook of Discrete and Computational Geometry', CRC Press, Boca Rotan, Florida, chapter 46, pp. 851–862.

Jensen, C. & Anderson, D. (1996), 'A review of numerically controlled methods for finish-sculptured-surface machining', *IEE Transactions* **28**(1), 30–39.

Jonker, P. & Vermeij, O. (1995), Mathematical morphology for 4D images: Skeletonization applied to path finding, *in* K. Wojciechowski, ed., 'Invited Lectures BENEFIT Summer School on Morphological Image and Signal Processing', Vol. 2, Silesian Technical University, Gliwice, Poland, pp. 109–118.

Joskowicz, L. & Taylor, R. (1996), 'Interference-free insertion of a solid body into a cavity: An algorithm and a medical application.', *The International Journal of Robotics Research* **15**(3), 211–229.

Kalpakjian, S. (1992), *Manufacturing engineering and technology, second edition*, Addison-Wesley, Reading, Massachusetts.

Kang, J.-K. & Suh, S.-H. (1997), 'Machinability and set-up orientation for five-axis numerically controlled machining of free surfaces', *The International Journal of Advanced Manufacturing Technology* **13**, 311–325.

Kaul, A. (1992), Minkowski sums: A simulation tool for CAD/CAM, *in* G. Gabriele, ed., 'Computers in Engineering - Volume 1', ASME, New York, pp. 447–456.

Kleiman, C., de Jager, J. & Lennings, L. (1997), Why use a concept modeler?, Technical Report K362, Delft University of Technology, Faculty of Industrial Design Engineering.

Ko, H., Kim, M.-S., Park, H.-G. & Kim, S.-W. (1994), 'Face sculpturing robot with recognition capability', *Computer-Aided Design* **26**(11), 814–821.

Kooijman, A. (1995), Srpost1.3 documentatie softwarebeschrijving, Technical Report K333, Delft University of Technology, Faculty of Industrial Design Engineering. In Dutch.

Kovács, Z. (1998), Integrating the free space generator with the sculpturing robot software, Technical report, Delft University of Technology, Faculty of Industrial Design Engineering.

Kruth, J. (1995), Rapid prototyping, a new application of physical and chemical processes for material accretion manufacturing, *in* 'Proc. International Symposium for ElectroMachining', EPFL, Lausanne, Switzerland, pp. 3–28.

Kuczogi, G. (1998), Collision avoidance for the sculpturing robot, Technical report, Delft University of Technology, Faculty of Industrial Design Engineering.

Latombe, J.-C. (1991), *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA.

Laur, D. & Hanrahan, P. (1991), 'Hierarchical splatting: a progressive refinement algorithm for volume rendering', *Computer Graphics* **25**(4), 285–288. (Proc. SIGGRAPH '91).

Lee, Y. (1997), 'Admissible tool orientation control of gouging avoidance for 5-axis complex surface machining', *Computer-Aided Design* **29**(7), 507–521.

Lee, Y. & Chang, T. (1995), '2-phase approach to global tool interference avoidance in 5-axis machining', *Computer-Aided Design* **27**(10), 715–729.

Lennings, A. (1997), Using rapid prototyping for freeform surface evaluation, *in* D. Roller, ed., 'Proc. 30th ISATA conf., dedicated on Mechatronics/Automative Electronics,', Automotive Automation Limited, Croydon, England, pp. 123–130.

Li, S. & Jerard, B. (1994), '5-axis machining of sculptured surfaces with a flat-end cutter', *Computer-Aided Design* **26**(3), 165–178.

Liu, X.-W. (1995), 'Five-axis NC cylindrical milling of sculptured surfaces', *Computer-Aided Design* **27**(12), 887–894.

Lozano-Pérez, T. (1983), 'Spatial planning: A configuration space approach', *IEEE Transactions on Computers* **C-32**(2), 108–120.

Magrab, E. B. (1997), *Integrated Product and Process Design and Development: the Product Realization Process*, CRC Press, Boca Raton, Florida.

Marshall, S. & Griffiths, J. (1994), 'A survey of cutter path construction techniques for milling machines', *Int. J. Prod. Res.* **32**(12), 2861–2877.

Menon, J., Marisa, R. & Zagajac, J. (1994), 'More powerful solid modeling through ray representations', *IEEE Computer Graphics and Applications* **14**(3), 22–35.

Morishige, K., Kase, K. & Takeuchi, Y. (1997), 'Collision-free tool path generation using 2-dimensional C-space for 5-axis control machining', *The International Journal of Advanced Manufacturing Technology* **13**, 393–400.

Mortenson, M. (1997), *Geometric Modeling*, second edn, John Wiley & Sons, New York.

Pahl, G. & Beitz, W. (1996), *Engineering Design : A Systematic Approach*, second edn, Springer-Verlag, London, UK.

Pugh, A. (1976), *Polyhedra: A Visual Approach*, University of California Press, Berkeley, California.

Pugh, S. (1996), *Creative Innovative Products Using Total Design*, Addison-Wesley, Reading, Massachusetts.

Ritter, X. & Wilson, J. (1996), *Handbook of Computer Vision Algorithms in Image Algebra*, CRC Press, Boca Rotan, Florida.

Rix, J. & Kress, H. (1997), Virtual prototyping - an open system environment to support the integrated product development process, *in* D. Roller & P. Brunet, eds, 'CAD systems development', Springer-Verlag, Berlin, Germany, pp. 106–118.

Roozenburg, N. & Eekels, J. (1995), *Product Design: Fundamentals and Methods*, John Wiley & Sons, Chichester, UK.

Saito, T. & Takahashi, T. (1991), 'NC machining with G-buffer method', *Computer Graphics* **25**(4), 207–216. (Proc. SIGGRAPH '91).

Serra, J. (1982), *Image Analysis and Mathematical Morphology*, Academic Press, New-York.

Spyridi, A. & Requicha, A. (1990), Accessibility analysis for the automatic inspection of parts, *in* R. Volz, ed., 'Proc. IEEE Int. Conf. Robotics and Automation', Cincinnatti, Ohio, pp. 1284–1289.

Tang, K., Woo, T. & Gan, J. (1992), 'Maximum intersection of spherical polygons and workpiece orientation for 4- and 5-axis machining', *Journal of Mechanical Design* **114**(3), 477–485.

Tangelder, J. (1993), Srplan1.1 user manual, Technical Report K294, Delft University of Technology.

Tangelder, J. & Vergeest, J. (1994), 'Robust NC path generation for rapid shape prototyping', *Journal of Design and Manufacturing* **4**, 281–292.

Tangelder, J., Vergeest, J. & Overmars, M. (1996), Computation of voxel maps containing tool access directions for machining free-form shapes, *in* J. McCarthy, ed., 'Proc. of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference', ASME, New York. CD-ROM.

Tangelder, J., Vergeest, J. & Overmars, M. (1997*a*), Freeform shape machining using Minkowski operations, *in* J.-D. Laumond & M. Overmars, eds, 'Algorithms for Robotic Motion and Manipulation', A.K. Peters, Wellesley, MA, pp. 301–310.

Tangelder, J., Vergeest, J. & Overmars, M. (1998), 'Interference-free NC machining using spatial planning and Minkowski operations', *Computer-Aided Design* **30**(4), 277–286.

Tangelder, J., Vergeest, J., van den Belt, H. & Overmars, M. (1997*b*), Producing physical prototypes using a sculpturing robot, *in* I. Horv'ath & A. Taleb-Bendiab, eds, 'Algorithms for Robotic Motion and Manipulation', The Manchester Metropolitan University, Manchester, UK, pp. 254–259.

Thomson, G. & Pridham, M. (1997), 'Controlled laser forming for rapid prototyping', *Rapid Prototyping Journal* **3**(4), 137–143.

Tovey, M. (1997), 'Styling and design : intuition and analysis in industrial design', *Design Studies* **18**(1), 5–31.

Ullman, D. (1992), *The mechanical design process*, McGraw-Hill, New-York.

Ulrich, K. T. & Eppinger, S. (1995), *Product Design and Development*, McGraw-Hill, New-York.

van den Belt, H. & Tangelder, J. (1997*a*), The direction selector software for the sculpturing robot, Technical Report K361, Delft University of Technology, Faculty of Industrial Design Engineering.

van den Belt, H. & Tangelder, J. (1997*b*), The orientation selector software for the sculpturing robot, Technical Report K373, Delft University of Technology, Faculty of Industrial Design Engineering.

van Dijk, C. (1994), Interactive Modeling of Transfinite Surfaces with Sketched Design Curves, PhD thesis, Delft University of Technology.

van Elsas, P. (1997), Free-form Displacement Features in Conceptual Shape Design, PhD thesis, Delft University of Technology.

van Vliet, L. & Verwer, B. (1988), 'A contour processing method for fast binary neighbourhood operations', *Pattern Recognition Letters* **7**, 27–36.

VDI (1987), VDI-richtlinien, IRDATA, VDI 2863, Technical report, Verein Deutscher Ingenieure, Dusseldorf. In German.

Vepsäläinen, A. (1990), An application of morphological filters to NC-programming, *in* P. Gader, ed., 'SPIE Vol. 1350 Image Algebra and Morphological Image Processing', SPIE - The international Society for Optical Engineering, Washington, pp. 177–183.

Vergeest, J. (1988), Sculpturing robot project, Technical Report K164, Delft University of Technology, Faculty of Industrial Design Engineering.

Vergeest, J. (1992), The sculpturing robot library, Technical Report K264, Delft University of Technology.

Vergeest, J., Broek, J., Schierbeek, B. & Tangelder, J. (1991), 'Prototyping of complex shapes', *International Journal of Systems Automation: Research and Applications* **1**(3), 305–324.

Verwer, B. (1990), A multiresolution work space, multiresolution configuration space approach to solve the path planning problem, *in* R. Volz, ed., 'Proc. IEEE Int. Conf. Robotics and Automation', Cincinnatti, Ohio, pp. 2107–2112.

Vincent, L. (1991), 'Morphological transformations of binary images with arbitrary structuring elements', *Signal Processing* **22**, 3–23.

Wall, M., Ulrich, K. & Flowers, W. (1992), 'Evaluating prototyping technologies for product design', *Research in Engineering Design* **3**, 163–177.

Walstra, W.H. Bronsvoort, W. & Vergeest, J. (1994), 'Interactive simulation of robot milling for rapid shape prototyping', *Computers and Graphics* **18**(6), 861–871.

Wernecke, J. (1994), *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, second edn, Addison-Wesley Publishing Company, Reading, MA.

Woo, T. (1994), 'Visibility maps and spherical algorithms', *Computer-Aided Design* **26**(1), 6–16.

Yagel, R., Cohen, D. & Kaufman, A. (1992), 'Normal estimation in 3D discrete space', *The Visual Computer* **8**, 278–291.

Yan, X. & Gu, P. (1996), 'A review of rapid prototyping technologies and systems', *Computer-Aided Design* **28**(4), 307–318.

You, C. & Chu, C. (1995), 'An automatic path generation method of NC rough cut machining from solid models', *Computers in Industry* **26**, 161–173.

You, C.-F. & Chu, C.-H. (1997), 'Tool-path verification in five-axis machining of sculptured surfaces', *The International Journal of Advanced Manufacturing Technology* **13**, 248–255.

Young, I., Peverini, R., Verbeek, P. & van Otterloo, P. (1981), 'A new implementation for the binary and Minkowski operators', *Computers Graphics and Image Processing* **17**, 189–210.

Curriculum Vitae

Hans Tangelder was born March 14, 1961 in Brunssum, the Netherlands. He obtained his Atheneum-B certificate in 1979 from the Rombouts College in Brunssum. In 1986 he received his MS degree in Mathematics from the Eindhoven University of Technology, The Netherlands. From 1986 to 1987 he was a software engineer with the Labour Force Survey Project at the Netherlands Central Bureau of Statistics. From 1987 to 1989 he worked as a software engineer in the Dutch Parallel Reduction Machine Project at the Computer Systems Department at the University of Amsterdam. From 1989 to 1998 he was a scientist at at Delft University of Technology, Subfaculty of Industrial Design Engineering and involved with the Sculpturing Robot project. In 1998 he joined the Subfaculty of Geodetic Engineering at the same university and is working as a scientist on measurement of complex 3-dimensional objects by fitting CAD models to images.