# Voxel Modelling for 

# Rapid Manufacturing 

Ronaldo Mercado



A Dissertation Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

De Montfort University

Leicester

February 2001

## Abstract

Rapid Manufacturing techniques create objects by adding material under computer control. The possibility of varying the material being added allows these processes to create Functionally Graded Materials. There are several research efforts that have succeeded in the creation of this type of objects but there are no established methods to model them in a CAD environment, since standard modelling applications presuppose a homogeneous object. This research explores the voxel modelling technique as a method to support Rapid Manufacturing where variable material composition will be possible.

Rapid Manufacturing processes are reviewed as well as applications of FGM objects, the decomposition model through voxels and the general CAGD modelling techniques. Alternative representation methods currently in research were reviewed and the representation of an F'GM using an FEA application was considered.

Visualisation techniques for the exploration of a voxel model are examined, including volume rendering. Visualisation software available for these operations is identified.

A system is developed based on the Visualization Toolkit (VTK), an open source, freely available visualisation library. Methods of generation of a voxel model, its visualisation and transfer to a Rapid Manufacturing machine are created. An example part was built based on a two-material model. The toolkit is extended to include the octree decomposition of graded material voxel models and the method is tested as a compression scheme, showing poor performance due to the overhead of pointers.

Despite its large memory requirements at high resolution, the voxel model seems suitable at the resolutions available through prospective creation methods.

## Acknowledgements

I would like to thank my parents and my family who have supported me all along. My special thanks to Prof. Blackledge and Prof. Dickens for recommending this research. I also owe thanks to all the special friends, colleagues and staff at De Montfort University.

Financial support for this research was received from De Montfort University and it is thankfully acknowledged.

## Contents

1 Introduction ..... 16
1.1 What is Rapid Manufacturing? ..... 16
1.2 Computational geometry and Voxel Modelling ..... 17
1.3 About this Thesis ..... 18
1.4 Structure of the Thesis ..... 20
1.5 Original contributions and publications ..... 22
2 Rapid Prototyping and Manufacturing ..... 24
2.1 Definition of rapid prototyping ..... 24
2.2 Stereolithography System ..... 25
2.2.1 3D Systems ..... 27
2.2.2 SOUP CMET ..... 27
2.2.3 D-MEC / SONY ..... 28
2.2.4 Light Sculpting Inc. ..... 28
2.3 Selective Laser Sintering (SLS) ..... 28
2.4 Solid Ground Curing (SGC) ..... 29
2.5 Laminated Manufacturing ..... 30
2.6 3-D Printing ..... 31
2.7 Jetting ..... 32
2.8 Fused Deposition Modelling (FDM) ..... 32
2.9 Shape Deposition Manufacturing (SDM) ..... 33
2.10 Rapid Tooling and Direct Metal Manufacturing ..... 33
2.11 Discussion ..... 35
3 Functionally Graded Materials (FGM) ..... 38
3.1 Applications ..... 39
3.1.1 Applications in space vehicles ..... 40
3.1.2 Application in stealth missiles ..... 43
3.1.3 Applications in aeroengines ..... 45
3.1.4 Application in diesel engines ..... 46
3.1.5 Applications in fuel burning systems ..... 48
3.1.6 Applications in integrated thermo-ionic/thermo-electric systems ..... 50
3.1.7 Applications in tungsten carbide cutting tools ..... 51
3.1.8 Applications in diamond cutting tools ..... 52
3.1.9 Applications for Graded Index Materials ..... 53
3.1.10 Applications in graded band gap semiconductors ..... 53
3.1.11 Application in biomaterials ..... 54
3.2 Creation methods ..... 56
3.3 Summary ..... 57
4 Geometric modelling ..... 59
4.1 Curve modelling ..... 60
4.1.1 Bézier curves ..... 60
4.1.2 B-Spline curves ..... 61
4.1.3 Rational Bézier and B-Spline curves ..... 63
4.1.4 NURB Algorithms ..... 66
4.2 Surface Modelling ..... 78
4.2.1 Tensor Product Bézier Surfaces ..... 78
4.2.2 B-Spline surfaces ..... 79
4.2.3 Rational B-Spline surfaces ..... 80
4.2.4 Rational Bézier patches ..... 82
4.2.5 Surface algorithms ..... 83
4.3 The IGES Standard ..... 83
4.3.1 IGES file structure ..... 83
4.3.2 Example specification: a NURBS curve and a NURBS surface ..... 85
4.4 Solid Modelling representations ..... 90
4.4.1 Boundary Representation Models ..... 90
4.4.2 Constructive Solid Geometry Models ..... 90
4.4.3 Decomposition models and voxel modelling ..... 91
4.5 The voxel model ..... 92
4.5.1 Memory requirements for the voxel model ..... 92
4.5.2 Compression Methods ..... 93
4.5.3 Manufacturing FGMs from a voxel model ..... 96
5 Modelling FGM objects ..... 98
5.1 Work at the University of Michigan ..... 99
5.1.1 Representation of heterogeneous objects ..... 99
5.1.2 Implementation of the $\mathrm{r}_{m}$-object representation ..... 102
5.1.3 The object model ..... 104
5.2 Work at the MIT ..... 105
5.3 Discussion of the methods ..... 106
6 Visualisation of voxel models ..... 108
6.1 Exploration of a voxel model ..... 108
6.2 Volume Rendering ..... 109
6.3 Visualisation software ..... 114
6.4 The Visualization Toolkit - VTK ..... 117
6.5 Design of the classes in the Visualization Toolkit ..... 118
7 Modelling FGMs ..... 122
7.1 Element discretisation approach ..... 122
7.2 Transforming a voxel model into a FEM grid ..... 122
7.3 Estimation of the properties ..... 123
7.4 Examples of modelling technique ..... 124
7.5 Discussion ..... 126
8 Software development ..... 129
8.1 VTK framework elements ..... 129
8.1.1 The VTK visualisation pipeline ..... 130
8.1.2 Data representation in VTK ..... 132
8.2 An extension to the toolkit ..... 136
8.2.1 The octree dataset ..... 137
8.2.2 Creation of the octree structure ..... 141
8.2.3 Test of memory usage and discussion ..... 143
9 Voxel model ..... 146
9.1 Generating the model ..... 146
9.1.1 VTK Datafile ..... 146
9.1.2 Semi automated model generation ..... 148
9.1.3 Pipeline code ..... 154
9.1.4 Discussion ..... 156
9.2 Visualisation pipeline ..... 156
9.2.1 Input step ..... 159
9.2.2 Classification step ..... 159
9.2.3 Contouring step ..... 160
9.2.4 Mapping ..... 160
9.2.5 Visualisation through slices ..... 162
9.3 Contouring for manufacturing ..... 162
9.4 Finite Element mesh generation ..... 164
9.5 Summary ..... 168
10 Summary, Conclusions and Future Work ..... 170
A Programs ..... 175
A. 1 Visualising a small voxel model ..... 175
A.1.1 Tcl script ..... 175
A.1.2 Example ..... 176
A. 2 Visualization of a model through slices ..... 177
A.2.1 Tcl script ..... 177
A.2.2 Example ..... 181
A. 3 Contouring a voxel model ..... 182
A.3.1 Tcl script ..... 182
A.3.2 Examples ..... 184
A. 4 Transformation into a voxel model ..... 184
A.4.1 Tcl script ..... 184
A.4.2 Example ..... 185
A. 5 Octree implementation ..... 186
B Object Oriented Concepts ..... 188
B. 1 Principles of object oriented software ..... 188
B. 2 Object Modeling Technique (OMT) ..... 190
C Results of the corner bracket example ..... 192
D Publication ..... 196
E Glossary ..... 214

## List of Figures

1.1 Concept application of FGMs ..... 19
1.2 System based on the voxel modelling technique ..... 20
2.1 Classification of Rapid Manufacturing processes ..... 25
2.2 The Stereolithography process ..... 26
2.3 SDC cutter ..... 31
2.4 The SDM process ..... 34
2.5 Advanced ALCOA injection moulding tool[18] ..... 35
3.1 Evolution towards FGM materials ..... 38
3.2 Types of graded structures ..... 39
3.3 Applications and potential applications of FGMs ..... 40
3.4 Schematic of composite combustion chamber ..... 42
3.5 Cross sectional schematic of a rocket engine ..... 44
3.6 Schematic of a TBC produced by electron beam-physical vapour deposition ..... 46
3.7 Schematic of an electron beam physical vapour deposition coater ..... 47
3.8 Micrograph of graded $\mathrm{Al}_{2} \mathrm{O}_{3}-\mathrm{YSZ}$ coating ..... 47
3.9 Schematic of a diesel engine showing location of TBCs ..... 48
3.10 Application of functional gradation of alloying Cr in a TiAl turbine blade. ..... 49
3.11 Schematic of a composite emitter electrode used in a thermo-ionic conversion system. ..... 51
3.12 Diagram of the interface in a cementless bone-prostheses fixation[45] ..... 55
3.13 Diagram of the interface in a PMMA cement bone-prostheses fixation and an IBBC fixation[45] ..... 56
4.1 A cubic Bézier curve ..... 61
4.2 Projection from homogeneous space to curve space ..... 65
4.3 Degree elevation process ..... 68
4.4 Graphical representation of the de Casteljau recursion ..... 71
4.5 Tensor product surface ..... 79
4.6 Example of an IGES file ..... 84
4.7 Structure of an IGES file ..... 85
4.8 IGES B-Spline surface ..... 87
4.9 Pixels and voxels ..... 91
4.10 Octree subdivision scheme. A three level representation of an object ..... 95
4.11 Scanning Electron Microscopy image of H 10 tool steel powder ..... 97
5.1 Computer representation of an $r_{m}$-object ..... 101
5.2 Architecture of the Heterogeneous Solid Modeler ..... 102
5.3 Design cycle using the heterogeneous solid modeller ..... 107
6.1 Transformation of medical data to create a physical 3-D model ..... 110
6.2 Surface contour created from a voxel model using VTK ..... 110
6.3 Volume rendering operations ..... 111
6.4 The effect of gradient operators: central difference gradient operator vs. in- termediate difference gradient operator ..... 113
6.5 Iris Explorer display of a visualisation pipeline ..... 116
6.6 Representation of a visualisation pipeline ..... 118
6.7 Visualisation of a combustor ..... 119
6.8 Visualisation of a height profile ..... 119
6.9 Distribution of the lines of code among the kits ..... 120
6.10 Distribution of the classes among the kits ..... 121
7.1 Geometry of the corner bracket modelled ..... 124
7.2 Boundary conditions for the model ..... 125
7.3 Operations performed in the corner bracket model ..... 127
7.4 Optimisation of material distribution ..... 128
8.1 Explicit vs. implicit network execution ..... 131
8.2 A VTK dataset ..... 132
8.3 Simple VTK cell types ..... 133
8.4 Dataset types ..... 135
8.5 Dataset object diagram ..... 136
8.6 The vtkOctree class in VTK's object hierarchy ..... 137
8.7 Object model for the octree ..... 138
8.8 The structured coordinates of a vtkStructuredPoints dataset ..... 139
8.9 Mapping of scalarId to values with vtkOctreeScalars ..... 139
9.1 Example VTK datafile ..... 147
9.2 Voxel model generated manually ..... 147
9.3 Alternative method of creation of a voxel model ..... 149
9.4 A simple geometry used to test the transformations ..... 150
9.5 Unigraphics presentation of the complete geometry of the turbine blade ..... 151
9.6 Unigraphics presentation of the simplified geometry of the turbine blade ..... 152
9.7 Algor presentation of the tetrahedral mesh generated ..... 153
9.8 Pipeline to transform an unstructured grid to voxels ..... 154
9.9 Topology and numbering conventions for a tetrahedral cell ..... 157
9.10 Topology and numbering conventions for a hexahedral cell ..... 157
9.11 Mimics display of a voxel model ..... 158
9.12 Visualisation of the two contour surfaces obtained from the voxel model ..... 161
9.13 Visualisation of the model through slices ..... 162
9.14 Contouring the voxel model to create STL files. Two-material model. ..... 163
9.15 Model manufactured from the voxel model ..... 164
9.16 Nodes for an element of the voxel model ..... 165
9.17 Programs for a system based on the voxel model ..... 168
10.1 Applications of the voxel model ..... 172
B. 1 Basic Object Model Notation ..... 191
C. 1 Stress distribution using one material (steel) ..... 193
C. 2 Material assignments ..... 194
C. 3 Stress distribution using five materials ..... 195

## List of Tables

2.1 Stereolithography based processes ..... 27
2.2 Laser Sintering processes ..... 29
2.3 Laminated Manufacturing processes ..... 30
2.4 Jetting equipment manufacturers ..... 32
2.5 FDM equipment manufacturers ..... 33
2.6 Indication of resolution achievable by the various RP processes ..... 37
4.1 Summary of the notation for Bézier curves ..... 61
4.2 Summary of the notation for B-spline curves ..... 63
4.3 Summary of the notation for rational B-spline curves ..... 66
4.4 Some entities in the IGES specification ..... 86
4.5 Quantities of data contained by various media ..... 93
7.1 Element sizes and corresponding node counts for the corner bracket example ..... 126
8.1 Memory usage of vtkStructured points and vtkOctree ..... 145
A. 1 Files for the octree implementation in VTK ..... 186
C. 1 Material properties used in the example bracket. Material 1 has the properties of steel and material 5 has the properties of aluminium . . . . . . . . . . . . . 192

## Chapter 1

## Introduction

### 1.1 What is Rapid Manufacturing?

Rapid Prototyping and Rapid Manufacturing are a set of manufacturing techniques for the creation of engineering parts under computer control. These techniques started in the late 80 s and have developed greatly over the last decade.

The term Rapid Manufacturing refers to the ability to produce small manufacturing runs of parts using the technique directly. The techniques are also collectively known as Rapid Prototyping (RP), in regard with their use in the production of prototypes. The terms Solid Freeform Fabrication (SFF), Layered Manufacturing and additive manufacturing are other names given to these techniques.

The development of these techniques occurred due to the combination of developments in many areas, such as polymers, laser techniques and computer technology. The first machine to be introduced was the SLA-1 by 3D Systems (California), based on the stereolithography process. The idea was already floating in industry and in the research community and there were several companies that pioneered the early developments, e.g. CMET in Japan or EOS in Germany.

There are several processes of Rapid Prototyping, such as stereolithography, selective laser sintering, laminated object manufacturing, fused deposition modelling, 3-D printing and jetting. All share a common characteristic of operation in a layer by layer fashion and
creating the objects by adding material instead of removing it. The process planning for parts built by adding layers is significantly simpler and it can be automated, resulting in significant time savings in the creation of a physical model from a design, hence the use of the techniques for the creation of prototypes.

Every process is linked to a characteristic machine and special materials. This is a limitation of the techniques, because in general it is highly likely that the material with the ideal properties for an application won't be available for the creation of a part in an RP machine.

There has been considerable work undertaken in the area of rapid prototyping. This has led to the subject of rapid tooling, where additive manufacturing techniques are being used to manufacture tools directly or indirectly. Rapid Tooling is a method of speeding the creation of parts while trying to overcome the limitation in materials that Rapid Manufacturing techniques have. The process is based in the time reduction possible in the very time consuming process of tool creation for series manufacturing processes such as injection moulding or die casting. In Rapid Tooling, RP techniques are used to produce mould parts for instance, and the final part is moulded in a suitable material. The next stage in this research will be to the use of additive manufacturing techniques to produce medium volumes of parts directly or indirectly. These techniques will probably be based on powder fusion or ink jet technology.

The next development has been the creation of objects with different materials in the volume and a graded composition in them. There are notable examples in the projects led by Prinz at Stanford and the Shape Deposition Manufacturing (SDM) process and Sachs et al. at the MIT using 3-D printing.

### 1.2 Computational geometry and Voxel Modelling

Computational geometry refers to the methods and mathematical methods used to represent geometry in a computer environment. Major modellers are available commercially implementing the ideas of geometry modelling: curve and surface modelling, a method well suited for the representation of free-form shapes and surfaces, and solid modelling. a method that emphasises and insists on a complete mathematical description of 'solid' objects and which
is suitable for algorithmic querying of a model.
One of the most simple methods to represent an object is through exhaustive enumeration, which consists in subdividing the space into regular cells and listing the occupancy of the cells in the model. The voxel modelling technique is a specific case of exhaustive enumeration based on cuboids. For this method, one limits the region of space that has to be represented and subdivides it as a three dimensional array. The most simple representation would only record a binary value to determine whether a region or cuboid is either interior or exterior to the solid model by attaching a "colour", either black (1, interior to the solid, filled) or white ( 0 , exterior to the solid, empty). For the representation of multiple materials, the memory usage increases with the number of "colours" that we allow in the model.

This type of geometric representation is similar in nature with that of 2-D images. The 3-D nature of it does have its peculiarities, but major methods used in 2-D imaging are applicable in 3-D volumetric models made up of voxels. The wealth of 2-D processing algorithms can be considered to be an advantage of the method. Also because of the regularity of the description method, algorithms for this representation tend to be simple, making it well-suited for parallel algorithms and hardware support.

Disadvantages are the poor resolution achievable because of the huge memory requirements. The large memory usage makes even small efficient algorithms perform poorly because of the number of cells on which the operations have to be performed. In general the voxel model is poor in terms of conciseness.

Many concepts from imaging can be translated to the voxel modelling, which could be regarded as a 3-D image representation or stacks of 2-D images piled up together. Much of what can be said about the advantages of vector graphics over raster graphics applies in the comparison of solid models over the voxel model. For example the voxel model suffers from the 'stair-step' effect and loss of accuracy when enlarged.

### 1.3 About this Thesis

The objective of this research thesis is to explore the possibilities of the voxel modelling technique to support rapid manufacturing techniques where variable material composition


Figure 1.1: Concept of a possible functionally graded material object built by Rapid Manufacturing
will be possible.
The motivation behind this project is the possibility to match the functional requirements demanded of a component at a given point by assigning localised material properties. This idea is not new, but the enhanced control and the complex geometries achievable using additive manufacturing open exciting possibilities. A concept application (figure 1.1) could be the construction of an engine block, which could have arbitrarily shaped cooling channels, with their walls built using a high thermal conductivity material to aid heat transfer, and a piston bore with hard material surface to prevent surface wear, while keeping a tough interior to withstand vibration. It is however necessary to identify possible applications because these are not yet clear. This project reviews applications of FGMs in various fields, showing uses in aeronautical engineering, astronautics and power conversion systems.

The resolution of additive manufacturing techniques to produce parts with an appropriate surface finish will require voxels of $5 \times 5 \times 5 \mu \mathrm{~m}^{3}$. This resolution involves heavy requirements of memory and computing power. Ideally the voxel modelling method would solve this problem by compression methods. One of the suggested compression methods, the octree approach, is implemented and tested within a visualisation toolkit/library in this project, proving to have too much overhead and therefore being unsuitable for the general use.

The issues of data transfer for the use of the model in a larger system are studied in order


Figure 1.2: System based on the voxel modelling technique
be able for instance to perform an analysis from a model (through Finite Element Analysis), to be able to visualise the model and also to manufacture parts based on the model. Figure 1.2 shows the model integrated within a system and how it would be used for supporting rapid manufacturing techniques.

One of the difficult aspects is also the visualisation of parts with variable composition of graded structures. The techniques required to visualise models voxel models, using simple sections and 3-D partially transparent models (volume rendering) are studied.

### 1.4 Structure of the Thesis

Chapter 1 is an introduction to the motivation and research objectives of the thesis.
Chapter 2 presents a classification of the principal methods of Rapid Manufacturing listing major manufacturers. A last section of this chapter presents Rapid Tooling, and Direct Metal Manufacturing.

Chapter 3 presents possible applications of functionally graded materials (FGMs) in science
and engineering. The chapter concludes with a short review of creation methods, including Rapid Manufacturing.

Chapter 4 presents the techniques used in Computer Aided Geometric Design (CAGD) for the representation of geometry. Major methods used in curve and surface modelling are presented, closing in on the most general of them, the Non Uniform Rational B-Splines (NURBS). A short look at the IGES standard and its way of representing NURBS, which was used for the transfer of geometry information later in the project. The chapter follows on to solid modelling techniques and finally focuses on voxel modelling.

Chapter 5 presents alternative representation methods for multiple material and materially graded objects currently in research. The work of Dutta et al. at the University of Michigan and Jackson et al. at the MIT are complete mathematical representation schemes that tackle these issues. The two methods are reviewed and discussed.

Chapter 6 looks at the issue of exploring a voxel model through visualisation techniques such as volume rendering. The visualisation pipeline for volume rendering is examined and some of the software applications available for volume rendering are surveyed. The Visualization Toolkit (VTK) is further examined, for its subsequent use.

Chapter 7 looks at a FGM representation approach using finite elements as used in the Finite Element Method. The method is tried in a simple structural analysis example with a commercial application package (ANSYS).

Chapter 8 focuses on the extension to the Visualization Toolkit through the implementation of an octree structure dataset for data compression, as suggested in chapter 4. The chapter looks at the elements of the VTK framework: the visualisation pipeline and data representation within the toolkit and indirectly some of the design philosophy behind it, based on the object oriented methodology. The extension is documented, tested and results are reported.

Chapter 9 examines further methods to use the voxel modelling technique for Rapid Manufacturing. The issue of generating a voxel model is examined and an application of the
visualisation tools (VTK) is presented to generate a model semi-automatically. Two examples for a simple geometry and a turbine blade are presented. Some other procedures are presented to process the voxel model for creation in a Rapid Manufacturing system.

Chapter 10 gives a summary, the conclusions and suggests future work.

The programs developed, concepts associated with the software development methodology used and a paper published are presented in the appendices.

### 1.5 Original contributions and publications

A paper was published in the Fourth International Scientific Colloquium CAx Techniques, Bielefeld 1999 and it is included in the appendix. The original contributions in this work are

- Reviews of background material relevant to the thesis, such as Rapid Manufacturing processes, applications of functionally graded materials (FGMs) and computational geometry methods used for computer representation of geometric entities (chapters 2, 3 and 4).
- A review of modelling methods currently in development for the solution of the issue of the representation of functionally graded material objects (chapter 5).
- A review of the possible use of visualisation software for voxel modelling and the survey of VTK as a voxel modelling framework (chapter 6).
- Generation of an FGM object model in a commercial Finite Element Analysis package (ANSYS) by discretising the space in cells with varying properties (chapter 7).
- An implementation of an octree decomposition for graded material voxel models and testing for the usefulness of the approach as a compression method (chapter 8).
- The development of data transfer methods and a method to integrate available tools to develop a system to support Rapid Manufacturing processes based on the voxel modelling method (chapter 9).
- The integration of tools available in the VTK toolkit to generate contour surfaces from a voxel model (chapter 9).
- The creation of an example part using the stereolithography method from a voxel model of two materials and the creation of a voxel model from a turbine blade geometry using the framework developed, based around the Visualization Toolkit (chapter 9).


## Chapter 2

## Rapid Manufacturing and Rapid Prototyping

### 2.1 Definition of rapid prototyping

Rapid Prototyping (RP) is generally understood as the process which involves the complete process of CAD modelling, data processing, transfer and building up the prototype layer by layer.

There has been considerable work and success in the area of rapid prototyping. This lead to Rapid Tooling, where additive manufacturing techniques are used to produce tools directly or indirectly. It has also lead to its use in manufacturing medium volumes of parts directly, hence the name Rapid Manufacturing for these techniques.

Other names given to these technologies are layered manufacturing, solid free-form fabrication (SFF), and automated additive fabrication[43].

RP systems enable users to produce prototypes quickly, efficiently and with a high degree of precision. The common characteristics of this family of processes are:

- Parts are automatically produced from CAE data sets. Many commercial RP machines can perform simulations to detect defects which come from the transformation of models to the industry standard STL format. This format consists of a list of vertices, triangles
and normals that determine the faces of a solid object. Specialised software (e.g. Magics[41]) allow editing, manipulating and visualising STL files.
- The techniques are additive: a solid object is built in a $2 \frac{1}{2} \mathrm{D}$ fashion by successively adding raw material.


Figure 2.1: Classification of Rapid Manufacturing processes

The approaches used to generate each single layer can be classified into three groups (see figure 2.1):

- Hardening of liquid materials (Stereolithography, Solid Ground Curing).
- Solid material layer addition (Laminate Object Modelling, Selective laser sintering, Fused Deposition Modelling, 3-D Printing).
- Generation out of gaseous phase (LASER Chemical Vapour Deposition).

The list of methods presented in this section is not exhaustive and only the major methods are listed, particularly the ones available within the facilities of the Rapid Manufacturing Group at De Montfort University.

### 2.2 Stereolithography System

Stereolithography was the first RP method to be invented. The first commercial stereolithography product, 3-D Systems SLA-1 was publicly introduced at the AUTOFACT Show in Detroit in November 1987[25]. The process is based on the use of photo-reactive polymers,
usually ones that react to ultraviolet light or short wave laser ( HeCd ). These resins solidify (polymerise) by absorbing sufficient irradiation energy. To allow the fabrication of parts, the SLA machine selectively polymerises one layer of the resin in a vat of the material. To create the next and subsequent layers, the object is dipped slightly deeper into the vat of liquid polymer. This process is repeated until the object is completed. A scheme of the process is shown in figure 2.2.


Figure 2.2: The Stereolithography process[15]

There are several manufacturers of machines which work on this principle. Table 2.1 lists a few companies that produce machines for this market.

Table 2.1: Stereolithography based processes

| Organisation | Country | Product | Web address |
| :---: | :---: | :--- | :---: |
| 3D Systems | USA | SLA 250, 500, 3500, | http://www.3dsystems.com |
|  |  | 5000,7000 |  |
| CMET(Mitsubishi) | Japan | SOUP | http://www.nttd-cmet.co.jp |
| D-MEC(Sony) | Japan | Solid Creation System <br> http://www.d-mec.co.jp <br> (SCS) |  |
| Aaroflex | USA | Solid Imager | http://www.aaroflex.com |
| Autostrade E-Dart | Japan | Solid Laser Plotter (SLP) | http://www.autostrade.co.jp |
| Light Sculpting Inc. | USA | LSI1212 | - |

### 2.2.1 3D Systems

3D Systems is the dominant company in the market. It produces a range of SLA systems of various envelope dimensions.

3D Systems machines' original deep dip, elevate and sweep process has been changed in more recent models with the Zephyr system, which utilises a vacuum-fed re-coating system[61]. In the Zephyr method, as opposed to the older re-coater blade system, the blade picks up resin from the vat and applies a thin layer as it sweeps across. This allows for a reduction in the time required to build parts. An additional advantage is the reduction of problems caused by trapped volumes - spaces that hold resin separate from that in the vat.

The resolution of a late SLA machine by 3D Systems (SLA 7000) is 0.0254 mm in the vertical direction and the laser spot diameter is 0.23 mm . These figures are however affected by the shrinkage of the parts, which depends on the materials, the build direction, the geometry and the curing process. Fitting parts created by SL machines still requires manual grinding and sanding of the interconnecting parts.

### 2.2.2 SOUP CMET

The Solid Ultraviolet Laser Plotting (SOUP) was developed by Mitsubishi Corporation in Japan and is marketed by CMET. The system is similar to that used in 3D Systems SLA machines', the main difference being that early models of the SOUP machine used an $x-y$
plotter arm to guide the laser. More recent models use the galvanomirror present in the SLA machines. In January 1999 a patent cross-license agreement was signed by 3D Systems with NTT DATA and NTT DATA CMET. By this agreement the companies have granted one another non-exclusive licenses to sell and produce stereolithography systems throughout the Asia-Pacific region. 3D Systems maintains an exclusive position in Europe and the United States[21].

### 2.2.3 D-MEC / SONY

Design Model Engineering Center (D-MEC) developed stereolithography machines known as Solid Creation Systems (SCS). The systems uses either HeCd or argon-ion laser and the laser beam spot size is also made adjustable. Depending on the machine, D-MEC systems offer large maximum working envelopes of up to $1000 \mathrm{~mm} \times 800 \mathrm{~mm} \times 500 \mathrm{~mm}$ [61]. This is almost double the envelope of the SLA 7000 made by 3D Systems.

### 2.2.4 Light Sculpting Inc.

The distinctive feature of Light Sculpting Inc.'s machine is the use of a light source that solidifies entire layers at once at a shorter distance through a mask. The resolution achieved by using masks is that of industrial printers - either 600 or 1200 dpi , compared with usual SLA machines which only reach 67 dpi. Due to the short irradiation distance, less expensive fluorescent or mercury bulbs can be used instead of laser[61].

### 2.3 Selective Laser Sintering (SLS)

This process was developed by Carl Deckard and Professor Joe Beaman at the University of Texas, Austin [61].

In the SLS process, a $\mathrm{CO}_{2}$ laser scans over a thin layer of powder to selectively fuse and join with other particles and form a solid mass. After whole cross section is scanned, the platform is lowered according to the specific layer thickness and a new layer of powder is spread on top. The process is then repeated.

Table 2.2: Laser Sintering processes

| Organisation | Country | Product | Web address |
| :---: | :---: | :--- | :---: |
| DTM | USA | Sinterstation 2500 | http://www.dtm-corp.com |
| EOS | Germany | EOSINT S, EOSINT M, | http://www.eos-gmbh.de |
|  |  | EOSINT P |  |

The companies that commercialise this process are DTM (USA) and EOS (Germany). The main difference in their machines lies in the powder delivery mechanism DTM machines use a roller levelling device, a powder tank located at one side of the powder bed and a container at the opposite side to collect redundant material. EOS uses a container that faces down toward the powder bed. EOS's design requires the feeding mechanism to travel only once over the powder bed.

The materials for this process are varied, e.g. thermoplastic, sand, elastomers, ceramic and metal powders[61].

### 2.4 Solid Ground Curing (SGC)

Cubital developed this Rapid Prototyping technique which is a variation on the stereolithography process. in the SGC method, a whole layer of photopolymer is solidified by UV light in a single run and cured completely. Unaffected resin is then vacuumed off. Wax support is then poured in and a milling tool removes excess material subsequently and levels the top surface for the next layer to be applied.

The machine has a high throughput and a relatively large envelope of up to 500 mm x $350 \mathrm{~mm} \times 500 \mathrm{~mm}(\mathrm{SGC} 5600)[14]$. It is possible to create parts overnight and no extra curing is required after they emerge. Also the use of wax means that no support structures are needed for overhangs and complex geometry. The wax is removed by melting or rinsing.

Table 2.3: Laminated Manufacturing processes

| Organisation | Country | Product | Web address |
| :---: | :---: | :---: | :---: |
| Helysys Inc. | USA | LOM | http://www.helysis.com |
| KIRA | Japan | KSC 50N and PLT A4 | - |
| Schoff Development Corp. | USA | SDC JP 5 System | http://www.sdcpro.com |

### 2.5 Laminated Manufacturing

Laminated manufacturing is a rapid prototyping technique that works on the principle of adding together several layers of material in sheet form. Table 2.3 lists a few processes in this category.

The LOM (Laminated Object Manufacturing) process, commercialised by Helysis Inc., uses laminated material coated with thermal adhesive that is glued successively layer by layer. Every layer is processed with a $\mathrm{CO}_{2}$ laser that cuts along the solid object's edge and cross-hatches the areas which don't belong to the solid. After all the layers have been added in the previous method, the cross hatched areas are easily removed by hand, to uncover the desired solid.

KIRA's paper lamination machines uses paper lamination technology to make 3-D models. Unlike Helysis's process, KIRA's doesn't require a pre-coated adhesive on the paper. The adhesive is applied as toner and glued thermally. The edges of the object are cut with a $x-y$ plotter knife instead of the $\mathrm{CO}_{2}$ laser used by Helysis' process. KIRA's machines can handle plain paper and toner (KSC-50N) and special paper too (PLT-A4).

Schroff Development Corporation commercialises what is possibly the cheapest RP system so far. The SDC JP 5 System prints the cross sections of every layer of the model. These cross sections are cut from laminated material. The sections are assembled by hand using positioning marks.


Figure 2.3: SDC cutter [58]

### 2.6 3-D Printing

3-D Printing is a Rapid Prototyping technique developed at the Massachusetts Institute of Technology by Prof. Emmanuel Sachs and Michael Cima[53]. 3-D Printing works on the principle of selectively spraying a binder on a powder bed. After a layer has been processed in this way, the powder bed is lowered, a new layer of material is added on top and the process is repeated. The region sprayed with the binder becomes part of the resulting solid object and the remaining powder is removed in a post processing step. The resulting part is known as the green part (about $50 \%$ dense), which is subsequently fired and infiltrated to make a dense metal part. Currently metal and ceramic parts are manufactured with this method, but there is the potential to manufacture multi-material parts[61]. MIT has licensed this technology to six companies to develop their own applications.

Table 2.4: Jetting equipment manufacturers

| Organisation | Country | Product | Web address |
| :---: | :---: | :---: | :---: |
| Sanders | USA | Modelmaker II | http://www.sanders-prototype.com |
| 3D Systems | USA | Thermojet | http://www.3dsystems.com |
| Objet Geometries | Israel | Objet Quadra | http://www.objet.co.il |

### 2.7 Jetting

Jetting technology works on the principle of ink jet printers, the important difference being that the material that comes out of the jet printing head(s) or nozzle(s) is melted wax or low melting point material instead of ink. This process associated with a controlled support bed enables the machine to build solid 3-D objects.

Table 2.4 lists three manufacturers of jetting machines. This technology is particularly appropriate for office application, among other things, because of the simple and reliable operation.

### 2.8 Fused Deposition Modelling (FDM)

Fused Deposition Modelling machines produce solid objects by dispensing successive layers of material through a robot-arm operated nozzle. Once a layer has been deposited, the support platform is lowered slightly a small distance determined by the thickness of the material deposited.

It is possible in this technique to use several nozzles for several materials to be deposited. One of the uses of several materials is the creation of support structures for a model. It is also possible to create multi-material objects with the combination of extruded materials.

The only manufacturer of commercial equipment in this category is Stratasys[7]. Stratasys's FDM machines have two nozzles available; one nozzle extrudes the structure material and the other extrudes the support material. The materials available for the process are $\mathrm{ABS}^{1}$, high impact grade ABS, investment casting wax and an elastomer.

[^0]Table 2.5: FDM equipment manufacturers

| Organisation | Country | Product | Web address |
| :---: | :---: | :---: | :---: |
| Stratasys | USA | FDM series | http://www.stratasys.com |

### 2.9 Shape Deposition Manufacturing (SDM)

Shape Deposition manufacturing is a layered manufacturing process that iteratively combines material removal and addition, as well as other intermediate processing operations performed on each layer.

It can be thought of being similar to FDM in that material is added through computer controlled nozzles that deliver material layer by layer. The distinctive steps in the SDM process are the material removal and stress relief steps (Figure 2.4).

Finger et al.[19] describe the process and explain its use for the creation of objects with embedded parts, and mention the technique in the production of wearable computer prototypes. The Shape Deposition Manufacturing process (SDM) has always had the ability to create multi-material objects. Further additions to this process, discussed by Fessler et al. [18], allow metallic powders from different powder feeders to mix under a laser. The result is an effective multiple graded material. Fessler's application was the creation of an advanced moulding tool (figure 2.5), that combines aluminium and stainless steel with a copper interior around the cooling channels. During the design, the computer representations used assumed a homogeneous material and the gradation was controlled by manipulating the process plan to deposit different materials[43]. This shows the lack of a suitable modelling method for functionally graded material objects.

### 2.10 Rapid Tooling and Direct Metal Manufacturing

The major methods presented in this section are still evolving and being extended by newer applications. SL, SLS, FDM and jetting are established techniques with proven practical applications.


Figure 2.4: The Shape Deposition Manufacturing (SDM) Process [19]

A limitation of RP techniques is that the materials that can be used, are restricted by the nature of the machines. This has lead to focusing RP efforts to the creation of tools in what is known as Rapid Tooling. Rapid Tooling enables pre-series manufacturing, used for testing and decision making. Some of the rapid tooling techniques are:

- Direct AIM and 3D Keltool (3D Systems) used for injection moulding.
- RapidSteel and RapidTool (DTM) used to create metal tools for moulding of plastics.
- PROMetal (Extrude Hone Corp.) also used for the creation of injection moulding dies, extrusion dies and metal components.


Figure 2.5: Advanced ALCOA injection moulding tool[18]

Another group of techniques driven by the demand for shortened product cycles is direct metal manufacturing, which does not require the use of intermediate binders, furnace densification or secondary infiltration. For example the Laser Engineering Net Shaping (LENS ${ }^{T M}$ ) process (Optomec Design Co.) is capable of producing metal parts directly. The system operates by combining a powder feed system and a laser focusing unit together in a nozzle. This process can directly process stainless steel, tool steel and titanium powders to produce near net shape metal parts.

Laser Powder Fusion, a method originally developed by Krupp in Essen is also capable of producing fully dense metal parts. The system works by scanning a high power $\mathrm{CO}_{2}$ laser on H10 tool steel powder. At De Montfort University a similar setup is being installed with the objective of processing functionally graded materials.

An analysis of the particle size and distribution for the tool steel powder used for Laser Powder Fusion experiments can give a representative idea of the maximum resolution achievable. The powder shows a bimodal particle size distribution (obtained by sieving) with peaks at $-150 \mu \mathrm{~m}+125 \mu \mathrm{~m}$ and $-90 \mu \mathrm{~m}+75 \mu \mathrm{~m}$.

### 2.11 Discussion

There have been significant advances in the field of Rapid Manufacturing since its beginnings over one decade ago, when the first SLA machine appeared.

The original processes have evolved into a family of related methods: stereolithography
(SL), selective laser sintering (SLS), fused deposition modelling (FDM), jetting and others. Several machines are offered by the various manufacturers to cater for the various uses and applications both in design bureaus and shop floors.

The limited number of available materials in RP processes is an important issue that led to the development of Rapid Tooling and Direct Metal Manufacturing processes.
$\mathrm{Su}[61]$ points out that considering the newly developed high-speed CNC machining processes, RP and RT processes loose out in speed, availability of materials and achievable tolerance. The distinct and unbeatable advantage of RP processes lies in the realisation of complex geometries with holes, overhangs and undercuts that are extremely difficult to make using conventional machining. Additionally being able to create parts with localised composition control, i.e objects with multiple materials and functionally graded material objects (FGM), with either discrete or continuous material variation is unique to additive processes.

These additive processes can also be seen as complementary, rather than competitive, and by combining Rapid Manufacturing techniques are able to build objects with overhangs and undercuts which are extremely difficult in conventional ways. An application in case are the conformal cooling channels in moulding tools, practicable using RP techniques, but whose complex geometry could not be created using standard machining (e.g. Fessler et al. ALCOA advanced moulding tool[18]).

The resolutions available for RP processes vary according to the material and process. Jetting technology is the most precise. Yet techniques that can be used for the creation of multiple material objects, such as selective laser sintering or direct powder fusion don't achieve high resolutions. Tolerances are often difficult to meet because of shrinkage or warping of parts subject to temperature gradients. An indication of available resolutions is given in table 2.6. Shrinkage can be a problem in RP. It almost never occurs uniformly because of the inhomogeneous distribution of temperature in the part. Although shrinkage can be compensated by enlargement of the CAD model to meet the tolerance, this procedure needs experience in material handling and it is prone to mistakes.

| Stereolithography | 0.23 mm (beam diameter) | 0.0254 |
| :--- | :--- | :--- |
| Light Sculpting | $0.0423 \mathrm{~mm}(600 \mathrm{dpi})$ | - |
| Selective laser sintering (SLS) | - | $0.05-1 \mathrm{~mm}$ |
| ThermoJet | $0.0846 \mathrm{~mm}(300 \mathrm{dpi})$ |  |
| Sanders Model Maker II | 0.07 mm | 0.013 mm to 0.076 mm |
| ObJet | $0.0423 \mathrm{~mm}(600 \mathrm{dpi})$ | $0.021 \mathrm{~mm}(1200 \mathrm{dpi})$ |

Table 2.6: Indication of resolution achievable by the various RP processes

## Chapter 3

## Functionally Graded Materials

## (FGM)

Functionally graded materials (FGM) refer to materials exhibiting spatially inhomogeneous structure and composition, resulting in corresponding changes in the properties of the material. FGMs do not present a sharp interface between constituent materials and typically present a graded change from one material to the other.

(a)

(b)

(c)

Figure 3.1: Illustrations of the evolution towards FGM materials[61]: (a) multi-material coated type object with sharp interface, (b) homogeneous composites and (c) FGM

Miyamoto et al.[45] mention that the whole concept of FGMs was first introduced along with composites in the seventies, but no actual investigation on how to design, fabricate and
evaluate graded structures took place until the 1980s. The name functionally graded material originated in Japan in the late 1980s.

Graded materials are not something new. For example case-hardened steel is a graded material developed long ago that is still in common use today. What is new is the realization that FGMs can be tailored at the micro-structural level to match specific functional requirements[54].

SFF processes can add on top of this 'micro-structural tailoring' the ability to produce arbitrary geometry under computer control, directly from a computer model.


Figure 3.2: Continuous (a) and stepwise (b) graded structures. Local gradients at the joint (c) and the surface (d). (Miyamoto et al.[45])

### 3.1 Applications

FGMs are a natural extension to the choice of materials available to a designer, when the various requirements in an application cannot be fulfilled by the use of conventional materials or composites. It is not surprising then that FGMs applications are those that require incompatible functions, e.g. chemical inertness and toughness, hardness with toughness, refractoriness and toughness.

Thermal barrier coatings (TBCs) are a very successful application of FGMs in the thermal protection of components. The benefits of a graded material in minimising thermal stresses
and varying thermal flux are explained by Markworth and Saunders[39, 40] who use the simple Voigt rule to estimate the material properties of the mixture and assume a quadratic material distribution. Their model shows that the heat flux varies with the shape of the distribution of materials and that the highest stresses usually occur at the high temperature surface for properties representative of ceramic and metal. There is the case, however, when the stresses achieve their highest values underneath the high temperature surface for certain quadratic material distributions.


Figure 3.3: Applications and potential applications of FGMs (Miyamoto et al.)

### 3.1.1 Applications in space vehicles

Space vehicles experience high temperatures when flying at high speed because of the aerodynamic heating caused by the friction with the atmosphere. The leading edge of vehicles flying at high speed, reaches radiant equilibrium temperatures above $2500^{\circ} \mathrm{C}$ For example the space shuttle, flying at $8 \mathrm{~km} / \mathrm{s}$ at an altitude of 120 km experiences $1500^{\circ} \mathrm{C}$ for a few minutes[45, p.249].

In the case of horizontally launched space vehicles, like the German Sänger program or the Japanese Single Stage to Orbit (SSTO), it is not only during reentry that long exposure to high temperatures happens. This occurs because these vehicles fly in the atmosphere at hyper-sonic speeds for a longer time than vehicles launched vertically by rockets, and the maximum heat is experienced during launch.

The thermal protection in the space shuttle is located in the nose, the leading edges and the rudder and it is composed of non-metallic carbon/carbon composites (C/C). Ceramic tiles can be used for temperatures up to $1200^{\circ} \mathrm{C}$.

A thermal barrier coating of $\mathrm{C} / \mathrm{C}$ composite coated with functionally graded $\mathrm{Si} / \mathrm{C}$ was developed and tested. A cone model was subjected for one minute to a supersonic (Mach 3) gas flow at $1900^{\circ} \mathrm{C}$ containing an amount of oxygen approximately equal to a standard atmosphere. The part composed of a $\mathrm{C} / \mathrm{C}$ substrate, a functionally gradient interface and an ungraded $100 \mu \mathrm{~m}$ thick $\mathrm{Si} / \mathrm{C}$ protective layer, showed "no discernible change in structure even after ten cycles"[45, p.249]. The cone models without the intermediate graded interface before the $\mathrm{Si} / \mathrm{C}$ coating deteriorated after the first cycle.

Rocket engines are another application of thermal barrier coatings (TBCs). A C/C combustion chamber with an $\mathrm{Si} / \mathrm{C}$ FGM protective layer was developed for HOPE, a Japanese space shuttle under development. A schematic of the engine is shown in figure 3.4. The walls of carbon/carbon composite were coated by a graded layer of $30 \mu \mathrm{~m}$ using chemical vapour infiltration (CVI) and subsequently by a second layer of $\mathrm{Si} / \mathrm{C} 100 \mu \mathrm{~m}$ thick using chemical vapour deposition (CVD). The tests showed that the FGM layer was very resistant to delamination and cracking. However the $\mathrm{Si} / \mathrm{C}$ layer showed de-lamination and corrosion after 500 seconds of stationary or pulsed combustion.

Other tests were done on rocket combustors using CVD-Si/C FGMs. The propellant used in the tests was nitrogen tetroxide (NTO) and monomethyl hydrazine (MMH) with firing cycles of 55 seconds with subsequent quenching by liquid nitrogen. After two test cycles no damage to the combustors was observed[45, p.250].

Rocket engines are a very hostile environment for the materials, due to the extremely high heat flux. Thermal barrier coatings of FGMs originally developed for turbine engine


Figure 3.4: Schematic of the carbon/carbon (C/C) composite combustion chamber for the engine of the reaction control system of the Japanese space shuttle, HOPE, with an FGM protective layer of silicon carbide/carbon ( $\mathrm{SiC} / \mathrm{C}$ ). The propellants are NTO (Nitrogen tetroxide: $\mathrm{N}_{2} \mathrm{O}_{4}$ ) and MMH (monomethylhydrazine: $\mathrm{N}_{2} \mathrm{H}_{3} \mathrm{CH}_{3}$ ). (Miyamoto et al.[45])
applications are used in rocket engines and protect the engine for much shorter work cycles but at higher temperatures and more severe thermal transients. A typical coating is a thin structure of 0.2 mm thickness. Large combustion chambers present such a high heat flux, that heat cannot be dissipated fast enough to prevent local hot spots and coating failure, and for these applications high conductivity copper is used to extract heat away from the chamber.

Thermal barrier coatings have also been used in liquid propelled rocket engines. Figure 3.5 shows potential locations for thermal barrier coatings (TBCs) in the high pressure hydrogen turbopump (left), main combustion chamber (centre), and the high pressure oxidiser turbopump (right). TBCs have been used as liners in the spark igniters and pre-burners, turbine housing liners, turbine blade shanks and vane shrouds.

On smaller regeneratively cooled thrust chambers for orbital manouvering systems, graded FGM thermal barrier coatings have also been used. The base layer of the graded parts were created by galvanoforming, depositing up to $25 \% \mathrm{ZrO}_{2}$ on a Ni metal chamber. This part is subsequently coated to $100 \% \mathrm{ZrO}_{2}$ by plasma spraying. The test of 550 seconds of combustion of combustion with this engine showed no de-lamination of the $\mathrm{ZrO}_{2}$.

### 3.1.2 Application in stealth missiles

The stealthiness of missiles and modern weapons depends on specific materials capable of absorbing emitted electro-magnetic energy to minimise reflected waves to enemy radars. Ceramic matrix composites with tailored microwave properties, reinforced with ceramic woven fabrics have been successful for these applications. The composite material offers greater toughness than monolithic ceramics, which are brittle.

The conducting properties of these ceramic composites varies with the material of the fibres, the matrix, the interfaces and the topology. Nasicon, with a structural formula $\mathrm{Na}_{1+x} \mathrm{Zr}_{2} \mathrm{Si}_{x} \mathrm{P}_{3-x} \mathrm{O}_{12}(0 \leq x \leq 3)$ has an electrical conductivity that varies by four orders of magnitude as a function of $x$. It is used to make ceramic composites with varying absorption of electro-magnetic waves.


Figure 3.5: Cross sectional schematic of a rocket engine.[45]

### 3.1.3 Applications in aeroengines

Graded thermal barrier coatings (TBCs) have many applications in aeroengines of both commercial and military aircraft and turbine engines in general. The principle in practice is that the higher the operating temperature of the engine, the higher the efficiency obtained. In order to increase the efficiency, gas inlet temperatures in a turboengine must be increased and the cooling of the parts must be decreased. Thus TBCs are located mainly on hot gas pathways, where thermal fatigue, temperatures and corrosion are critical. The thickness of the coatings on these paths is usually thin ( $<0.4 \mathrm{~mm}$ ) to prevent spalling. but thicker coatings can be used in other sections of the engine, e.g. seals.

Turbine and engine coatings are also subject to high corrosion and erosion from particles.
The two methods used to create thermal barrier coatings in aeroengines are electron beam-physical vapour deposition (EB-PVD) and plasma spraying.

EB-PVD is used for coatings on the air-foils of blades and vanes. These are thin coatings as shown in figure 3.8. The apparatus used to create the coatings is schematically shown in figure 3.7. The bonding between the ceramic TBC and the metallic super-alloy in a turbine blade core is done using a single layer bond coat (thin metallic bond coat) of either NiCrAlY, NiCoCrAlY or Pt -Al. The use of either NiCrAlY or NiCoCrAlY presents two problems:

- At the metallic interface, it is desirable to have a minimum diffusion of Cr and Al in the super-alloy.
- At the ceramic interface, it is desirable to have as high as possible concentration of Cr and Al , to build up a dense, stable, protective alumina $\left(\mathrm{Al}_{2} \mathrm{O}_{3}\right)$ scale.

To solve the problem at the metallic interface, it is possible to increase diffusion barrier elements (platinum, palladium) or reduce the $\mathrm{Cr}, \mathrm{Al}$ at the interface. At the ceramic interface, the solution is to increase the oxide forming Cr and Al. The optimal concentration distribution could be met with a graded structure with varying content of Al and Cr .

There are several good characteristics of TBC produced by EB-PVD.

- Smooth surfaces without further polishing


Figure 3.6: Schematic of a thermal barrier coating (TBC) produced by electron beam-physical vapour deposition. The bond coat is graded.

- Good erosion resistance
- No closure of cooling holes
- Outstanding resistance to thermal shock, due to the columnar micro-structure

These characteristics lead to a considerably extended lifetime.
Coatings produced by plasma spraying are used in inside liners of combustors where the fuel ignites with air, and on the platforms of turbine vanes and blades, where the hot gases expand into the turbine section. Thicker coatings $(2.5 \mathrm{~mm})$ created by plasma spraying are used for abradable blade outer air seals. Military aircraft aeroengines use TBCs in augmentor (afterburner) components (tail cones, flame holders, heat shields and duct liners) which are not present in commercial aircraft.

### 3.1.4 Application in diesel engines

Diesel engines have also benefited from the use of functionally graded thermal barrier coatings. TBCs have been applied on piston crowns, valve faces and cylinder heads. Experimental TBCs have been tested on cylinder liners, exhaust valve systems and valve seats. The advantages obtained by using TBCs are:

- increased power density,


Figure 3.7: Schematic of an electron beam physical vapour deposition coater [45, p.195]


Figure 3.8: Micrograph of graded alumina-yttria stabilised zirconia ( $\mathrm{Al}_{2} \mathrm{O}_{3}-\mathrm{YSZ}$ ) coating. The columnar micro-structure provides outstanding thermal shock resistance[45, p.196]


Figure 3.9: Schematic of a diesel engine showing location of TBCs[45, p.256]

- reduced heat loss,
- reduced fuel consumption,
- reduced exhaust emissions

It has been shown that $5 \%$ reduction in fuel consumption is obtained by insulating the combustion chamber with 2 mm thick functionally graded $\mathrm{TBCs}[45, \mathrm{p} .255]$.

### 3.1.5 Applications in fuel burning systems

Miyamoto describes two applications in fuel burning systems: turbine blades of titanium aluminide and porous silicone carbide ceramic liquid fuel evaporator tubes with tailored


Figure 3.10: Application of functional gradation of alloying Cr in a TiAl turbine blade. porosity.

Turbine blades are usually made of heavy super-alloys and an interesting prospect is the use of lighter materials, such as $\gamma$-titanium aluminide. This is a suitable material at intermediate temperatures $\left(600^{\circ} \mathrm{C}\right.$ to $\left.800^{\circ} \mathrm{C}\right)$. Unfortunately the creep strength and the ductility are two opposite properties in this material. While the $\alpha$ phase with heat treatment has good creep strength, its ductility is poor. The $\alpha-\beta$ two-phase field with heat treatment presents acceptable creep strength and low but acceptable ductility. The desired gradient of properties can be obtained by changing the concentration of alloying Cr in the TiAl. The effect of Cr is a change in the equilibrium volume ratio of the $\alpha+\beta$ phase during isothermal annealing. If sufficient Cr is present, a fully lamellar micro-structure develops, with excellent creep strength. Turbine blades of titanium aluminide with gradients in Cr content have been produced by hot isostatic pressing[45, p.257].

Fuel evaporator tubes are used to premix air and fuel before combustion. This pre-mixing achieves optimised fuel efficiency at low emission levels of soot, hydrocarbon and nitrogen oxide gases. The evaporation surface is the exterior surface of the tubes, while the interior of the tube is where the combustion takes place. The porosity of the tube must vary from the interior, where porosity is to be avoided, to the exterior, where porosity is advantageous for the evaporation. Porous silicon carbide ceramic tubes can be made with a continuous graded
function. The gradation of the material can also reduce the probability of failure, from the thermal stress generated by a high temperature gradient $-1500^{\circ} \mathrm{C}$ at the inner and $550^{\circ} \mathrm{C}$ at the outer tube wall[45, p.258].

### 3.1.6 Applications in integrated thermo-ionic/thermo-electric systems

A high efficiency hybrid energy conversion system (HYDECS), developed as part of the second Japanese FGM program, shows several applications of functionally graded materials. The system has a solar receiver system, a thermo-ionic energy conversion step, for temperatures at around 2000 K , a thermoelectric energy conversion unit at temperatures around 1100 K and a heat radiator at around 300 K .

The solar receiver system is a C/C composite heated to temperatures around 1900 K at its bottom transmitting plane. The system applies functional gradation in both the fibre volume fraction, which increases toward the central axis of the cavity, and the fibre orientation, which aligns fibres in the direction of the desired heat flow. The orientation of the fibres is axial in the central areas of the cavity and more radial toward the outer edges of the collector. The use of functionally graded materials allows an increase in $100^{\circ} \mathrm{K}$ to $150^{\circ} \mathrm{K}$ at the transmitting planar bottom surface[45, p.260].

Thermo-ionic conversion operates on the principle of electrons discharged from a hot emitter and collected at a lower temperature. The material used for the emitter is rhenium (Re) and the material of the heat receiving plate is titanium carbide ( TiC ). To join these two plates together, an advantageous gradation of $\mathrm{TiC} / \mathrm{Mo}$, MoW and WRe was developed. The characteristics that make this gradient plate convenient are:

- Excellent heat conductivity,
- Reduction of the thermal stresses among the plates,
- Diffusive barrier action between the TiC (heat receiving) and the Re (emitting) plates. The collector is made of sputtered niobium oxide $\left(\mathrm{NbO}_{x}\right)$ on a molybdenum (Mo) electrode. The thermo-ionic conversion system built using these materials was operated at emitter-


Figure 3.11: Schematic of a composite emitter electrode used in a thermo-ionic conversion system.
collector temperatures of $1600^{\circ} \mathrm{C}-760^{\circ} \mathrm{C}$, with cesium reservoir temperatures of $330^{\circ} \mathrm{C}$ and a maximum output of $80 \mathrm{KW} / \mathrm{m}^{2} .15^{\circ} \mathrm{C}[45$, p.261].

The thermoelectric conversion benefits from the use of a gradation of the dopant in the base compound. The materials selected for the conversion units are silicon germanium compound $\mathrm{Si}_{.8} \mathrm{Ge}_{.2}$ for the higher temperature range of 1300 K to 900 K , lead telluride ( PbTe ) for the intermediate temperature range of 900 K to 500 K and bismuth telluride $\mathrm{Bi}_{2} \mathrm{Te}_{3}$ for the lower temperature range of 500 K to 300 K . This selection is based in the thermoelectric figure of merit for the various materials, which is a function of the temperature, the nature of the carriers and their concentration. It has been estimated that the effective maximum power (the figure of merit) for a n-type lead telluride conversion unit can be optimised by grading the concentration of the dopant lead iodide $\left(\mathrm{PbI}_{2}\right)$. Similarly, "a conversion unit made of an n-type SiGe FGM with gradation in the concentration of the phosphorus dopant shows a marked improvement in output power characteristics" [45, p.264].

### 3.1.7 Applications in tungsten carbide cutting tools

A typical cutting tool of tungsten carbide (WC) is made by sintering powders at high temperatures with cobalt (Co) as binder. The hardness of the resulting tool depends on the percentage of the binder and on the grain size of the WC. The control of the grain size has been achieved by controlling the atmosphere and the rates of heating and cooling during the liquid sintering phase of the process. Sumitomo Electric Industries Ltd. has developed functionally graded cutting tools using these methods. The principles in practice are that
the hardness of the cutting tool decreases with increasing binder content. At constant binder content, the hardness of the tool decreases with increasing grain size. The rupture strength and fracture toughness of the tools decreases with increasing hardness, almost irrespective of grain size or binder content.

By controlling the process parameters, heating and cooling rates, a WC/Co throw away chip was developed that presents a varying concentration of Co from the surface to the interior. The result is that the surface harder than the interior. The outer surface is almost completely ceramic without metal binder, which has high hardness and high surface compressive stress. The WC/Co cutting tools are subsequently coated by chemical vapour deposition with a layer of titanium nitride TiN , a layer of alumina $\left(\mathrm{Al}_{2} \mathrm{O}_{3}\right)$ and a layer of titanium carbonitride. "The high surface hardness and compressive stress plus the toughness of the interior almost doubles the wear resistance, and increases the tool life as much as fivefold compared with conventional cermet tools." [45, p.273]

These FGM multiply coated WC/Co throw away chips are also very resistant to flank wear and allow for high machining speeds and high feed rates. Additionally the graded layers permit better control of the thermal stresses which arise due to the unmatched thermal expansion rates of the metal and the ceramic.

### 3.1.8 Applications in diamond cutting tools

Diamond cutting tools are used for high precision machining of soft components. To create a tool, a diamond crystal is joined to a metallic alloy shank using a silver solder. One disadvantage of the silver solder is that its lack of stiffness causes vibration and loss of machining accuracy. To solve this problem an extremely stiff FGM diamond tool was developed. The tool is made up of three layers, one of diamond crystal, a graded diamond/SiC layer and a SiC shank. The graded layers vary from 0 to $80 \%$ in volume of diamond powder with polymer binder. Additionally the graded layer reduces the thermal stresses in the tool, and it is estimated that the life of the tool can be extended by $30 \%$. Unfortunately this tool is still too expensive to manufacture to be competitive commercially.

### 3.1.9 Applications for Graded Index Materials

The applications of a continuous variation of the refractive index in a medium has been explored. The media is collectively known as GRIN for graded index or gradient index. There are three main types of graded index (GRIN) lenses: axial, radial and spherical depending on the distribution of the refractive index. Light in a radial GRIN lens with a quadratic refractive index distribution

$$
n(r)=n_{0}\left(1-\frac{1}{2} A r^{2}\right)
$$

follows a sinusoidal path in which every ray passes through one point at a distance $L / 2$, forming an inverted image and again at a distance $L$, forming an upright image, where $L=\frac{2 \pi}{\sqrt{A}}$ and $A$ is a positive distribution constant. Radial GRIN lenses of this kind have been used as connectors and couplers for optical fibres and as imaging lens arrays in photocopiers[45, p.290].

Glass fibres used in data communications currently use a single-mode step-index fibre, which offers superior data-carrying performance. Other possible types of glass fibres are multi-mode step-index fibre and multi-mode graded-index fibre. While in the multi-mode step-index fibre there are time differences among the various modes (wavelengths), in a graded-index fibre with an optimised profile all modes propagate at the same velocity. This means that an impulse is not spread over time and a significant increase in the data-carrying capacity can be achieved[45, p.292].

Following a similar idea using polymers instead of glass, polymer optical fibre (POF) has been considered for short-distance communication applications such as local area networks (LANs). For the applications that will be required in the near future, the bandwidth offered by step-index (SI) POF will not be enough, therefore graded-index POF (GI POF) have been considered[45, p.296].

### 3.1.10 Applications in graded band gap semiconductors

Semiconductor heterojunctions using graded materials have been considered for some electronic applications. In the case of bipolar transistors, the application of graded bandgap
structures offers unique energy band profiles with improved characteristics[45, p.286].
A crystallographic function of a graded structure is the gradual introduction of misfit dislocations in a thick buffer layer. This procedure is used to grow heterostructures using epitaxial growth on substrates with non-matching lattice constant. This has been used in orange-coloured light-emitting diodes[45, p.284].

The use of a quasi-field effect for graded structures has been proposed. Based on this effect, it should be possible to control the behaviour of carriers.

Another application mentioned is the removal of potential barriers for carriers at heterojunctions by gradual change in the composition of the alloy. This reduction at a very small scale leads to quantum size effects that allow the design of a variety of wave functions and densities of state[45, p.286]. Examples of these applications are high electron mobility transistors (HEMT) and quantum well lasers.

An application in semiconductor lasers (single quantum well lasers) is the improvement of the separate confinement heterostructures ( SCH ) which have abrupt changes in the energy band profiles. A graded index (GRIN) SCH laser has reduced photo-absorption and enhanced carrier capture[45, p.288].

### 3.1.11 Application in biomaterials

The Interface Bioactive Bone Cementation (IBBC) is a technique that combines the advantages of two other techniques: bone cementation using polymethyl methacrylate (PMMA) and bioactive binding using hydroxyapatite (HAp), a bioactive calcium phosphate ceramic. This graded interface for bone orthopedic implants is in use in Japan since 1985.

The bone can be fixed to the prostheses in several ways. The cementless fixation shown in figure 3.12 works by inserting the prostheses tightly into the bone, which is reamed to the same shape of the insert. A live soft tissue layer grows in between the component and the bone. Weight bearing and walking may cause pain in this configuration, and worse, the micro-motion may loosen the binding.

An improvement on the cementless binding method is achieved by coating the metallic titanium alloys with a bioactive ceramic layer (HAp layer) of 50 to $100 \mu \mathrm{~m}$ that provides


Figure 3.12: Diagram of the interface in a cementless bone-prostheses fixation[45]
physico chemical bonding. The bonding is improved more by making the surface of the metal porous. The optimum pore size is 300 to $600 \mu \mathrm{~m}$. Bone growth in the pore cavities provides firmer mechanical bonding, but pain may still happen because of micro motion and small spaces between the bone tissue and the beads.

The conventional technique of using PMMA bone cement to join the implant to the bone is advantageous in that the prostheses can be completely fixed in the bone immediately after surgery, since the cement hardens in minutes after its components are mixed and kneaded. A problem of the method, though is that over the time soft living tissue can become interposed between the bone and the bone cement (figure 3.13). The IBBC method improves on this by applying one to three layers of HAp granules between the bone and the bone cement. The inclusion of HAp granules in the region between the cement and the bone promotes bone ingrowth and the HAp granules chemically bond to the bone.

Fixation with Bone Cement
(PMMA)
Bone Cement Fixation
Interface Bioactive Bone Cement Fixation

Figure 3.13: Diagram of the interface in a PMMA cement bone-prostheses fixation and an IBBC fixation[45]

### 3.2 Creation methods

A classification proposed in [45] includes:

- Bulk processing. This includes processes that create FGMs from powder stacking, powder sintering and hot pressing.
- Layer processing, which includes spray deposition, laser cladding, vapour deposition and deposition by electro-transport.
- Pre-form processing, which refers to FGMs created through solid state and liquid phase diffusion or processing a material to change its properties (e.g. porosity) inhomogeneously by submitting it to thermal or electric fields.
- Melt processing, which refers to settling of grains in molten materials (e.g. W in a W-Fe-Ni melt[45, p.213]) under plain gravity or using centrifugal forces.
- Joining, like low temperature solid-state joining, transient liquid phase joining or liquid
phase joining.
SFF processes are also presented in the exposition of methods to create FGMs[45, p. 220-232].
- Laminated object manufacturing has been used to create FGMs by substituting the standard paper in sheets with tape-cast, flat sheets consisting of fine ceramic or metal particles dispersed in a polymer matrix.
- Stereolithography was used to create ceramic filled polymer parts and it could potentially be used to create varying composition parts by filling the liquid polymers with two or more different materials.
- Selective Laser Sintering (SLS) and 3-D Printing are also mentioned as possible FGM creation processes.
- Based on the SFF process of fused deposition modelling (FDM), the deposition of ceramics and metals have been demonstrated successfully.
- Another SFF process, Extrusion Freeform Fabrication, similar in nature to FDM is reported to have been used to fabricate FGMs by depositing layers of thermo-plastics using a computer controlled extrusion head. The creation of FGMs was possible by using two extruders to dispense different materials in a small mixing head. Ceramic and metal powders were used in the fabrication experiments.

Fessler et al. have also created FGMs through the Shape Deposition Manufacturing (SDM) process. Their product was an advanced moulding tool with a graded transition from aluminium to stainless steel[18]. Jepson et al.[27] have created small FGM tungsten carbide and cobalt dies through Multi Material Selective Laser Sintering (M2 SLS).

### 3.3 Summary

The concept of functionally graded materials has been present in science and engineering from long ago. The difficulty in controlling the material composition in the volume and the
difficult procedures of creation have not stoped newer and more ingenious applications from being developed.

It is usually in the very advanced applications where "normal" engineering techniques still haven't proved sufficient that we find applications for FGMs. A large number of applications have been developed from space exploration programs, where elements are subject to extreme temperatures and extreme thermal stresses. Thermal barrier coatings (TBCs) of ceramic/metal are a typical case.

Some Rapid Prototyping and Manufacturing techniques can control the material composition and can be used for the creation of FGM parts. However these parts have still not been modelled. In previous work[43] it was shown that realisation of parts is ahead of modelling. This is still the case, although we'll see in chapter 5 and chapter 9 techniques that intend to change this situation.

## Chapter 4

## Geometric modelling

The computer representation of surfaces, curves, objects and assemblies requires models to capture infinite point information in finite storage. The completeness of the models used has been driven by applications. The aim of the computer representations is ultimately to capture enough information to facilitate and to automate the processing of design information. This need has led over the years to the creation of complex product models of which the geometry model is a subset.

The major approaches for geometric modelling representations are surface modelling and solid modelling. Surface models are better suited for the representation of complex surfaces; solid models provide a complete, unambiguous representations of solids. The beginnings of curve and surface modelling with computational geometry applications can be attributed to early works of P. Casteljau at Citroën and P. Bézier at Rénault in the 1960's. Research in solid modelling emerged in the 1970's from early exploratory efforts that sought shape representations suitable for machine vision and for the automation of tasks performed by designers and engineers[55].

Naturally, the use of geometric models is not restricted to CAD/CAM/CAE applications and there are interdisciplinary cooperations and overlaps with physics, geo-science, computer graphics and several other fields.

Curve and surface modelling techniques provide on mathematical methods to represent geometry. They are used in major modelling programs and in many design applications some-
times far distanced from Computer Aided Geometric Design (CAGD). Even simple drawing applications provide facilities for the user to draw simple curves and sometimes Bézier curves or B-splines. This chapter summarises the main algorithms used in curve and surface modelling with emphasis in Non Uniform Rational B-Splines (NURBS), which are the most general method to represent curves and surfaces and encompass simpler forms and provide representation for standard analytical shapes. References [16],[52] provide comprehensive coverage of algorithms and methods in curve and surface modelling.

When surveying the literature on curve and surface modelling, one notices the notation differences among authors, and different approaches to counting and designating functions. In general the notation followed in this chapter corresponds to reference [4].

### 4.1 Curve modelling

### 4.1.1 Bézier curves

A Bézier curve is defined as a parametric curve in space with the following formulation:

$$
\begin{equation*}
\mathbf{p}(t)=\sum_{i=0}^{m} \mathbf{b}_{i} B_{i}^{m}(t) \quad t \in[0,1] . \tag{4.1}
\end{equation*}
$$

This formulation is based on the idea of a set of $m+1$ control points $\mathbf{b}_{i}$ and $m+1$ BernsteinBézier basis functions $B_{i}^{m}(t)$. The basis functions are defined as:

$$
B_{i}^{m}=\binom{m}{i} t^{i}(1-t)^{m-i}
$$

The control points $\mathbf{b}_{i}$ in equation 4.1 form a control polygon. There are several notable properties of these curves that made them a suitable choice for design, one of the most important being that the curves are invariant under affine transformations e.g. rotation or scaling of the control points. Another very appealing property of the Bézier curves is that the control points have a direct geometric meaning in relation to the curve being modelled, namely that the first control point coincides with the beginning of the curve and that the last control point coincides with the end, while the second and next to last control points


Figure 4.1: A cubic Bézier curve. The control points have immediate geometric meaning.

| Element | Symbol |
| :--- | :--- |
| Bézier curve | $\mathbf{p}(t)$ |
| Basis function | $B_{i}^{m}(t)$ |
| Degree of the curve | $m$ |
| Control points | $\mathbf{b}_{i} \quad i=0, \ldots, m$ |
| No of control points | $m+1$ |

Table 4.1: Summary of the notation for Bézier curves
show the direction of tangent of the curve at the beginning and end of the curve.
These geometric properties made it a choice for many systems to implement Bézier curves as a standard method of drawing 2-D curves, notably in the Windows graphics device interface (GDI)[13].

### 4.1.2 B-Spline curves

The parametric formulation of a B-Spline curve in three dimensions is:

$$
\begin{equation*}
\mathbf{p}(u)=\sum_{i=1}^{n} \mathbf{d}_{i} N_{i, k}(u), \quad u \in\left[u_{k}, u_{n+1}\right] \tag{4.2}
\end{equation*}
$$

The curve is defined for the parameter $u$; there are $n$ control points that multiply $n$ basis functions $N_{1, k}, N_{2, k} \ldots, N_{n, k}$ of order $k$ defined over a knot set $\{u\}$. The number of knots
in the knot set depends on the order of the B-Spline and the number of control points, i.e. $\{u\}=\left\{u_{1}, u_{2}, \ldots, u_{n+k}\right\}$.

The basis functions are defined through a recursive formula:

$$
\begin{gather*}
N_{i, 1}= \begin{cases}1 & u_{i} \leq u \leq u_{i+1} \\
0 & \text { otherwise }\end{cases} \\
N_{i, k}(u)=\frac{u-u_{i}}{u_{i+k-1}-u_{i}} N_{i, k-1}(u)+\frac{u_{i+k}-u}{u_{i+k}-u_{i+1}} N_{i+1, k-1}(u), \tag{4.3}
\end{gather*}
$$

B-Splines and Bézier curves can both represent the same curves, and algorithms exist that can transform one representation to the other. A Bézier curve is a B-Spline defined over a special knot set of the form:

$$
U=(\underbrace{0,0, \ldots, 0}_{k \text { times }}, \underbrace{1,1, \ldots, 1}_{k \text { times }})
$$

where $k$ is the order of the curve. This knot set is obtained by inserting extra knots where necessary and splitting and rescaling the curve to the standard Bézier parameter value range $u \in[0,1]$. The knot insertion algorithm is given in section 4.1.4.

It is worth noting that the shape of a B-spline does depend on the knot set chosen. Piegl and Tiller[52] restrict their definition of a B-spline by stating that the knot set must be of the form

$$
U=\{\underbrace{a, \ldots, a}_{k}, u_{k+1}, \ldots, u_{n}, \underbrace{b, \ldots, b}_{k}\}
$$

which results in the endpoint interpolation property:

$$
\begin{equation*}
\mathbf{P}(a)=\mathbf{d}_{1} \quad \text { and } \quad \mathbf{P}(b)=\mathbf{d}_{n} \tag{4.4}
\end{equation*}
$$

This restriction is not adopted by all authors, allowing forms such as the uniform B-splines which are defined over an uniformly spaced knot set. Property 4.4 does not apply for a uniform knot set.

| Element | Symbol |
| :--- | :--- |
| B-Spline curve | $\mathbf{p}(u)$ |
| Basis function | $N_{i, k}(u)$ |
| Order of the curve | $k$ |
| Control points | $\mathbf{d}_{i} \quad i=1, \ldots, n$ |
| No of control points | $n$ |
| Knots | $u_{i} \quad i=1, \ldots, n+k$ |
| No of knots | $n+k$ |

Table 4.2: Summary of the notation for B-spline curves
The multiplicity of a knot in the knot set is linked to the number of continuity conditions at that knot by the relation:
number of continuity conditions at breakpoint $\xi+$ number of knots at $\xi=k$,
where $k$ is the order of the B -spline. For a B -spline of order $k$, it is therefore only useful to have at most multiplicity $k$ for any particular knot.

### 4.1.3 Rational Bézier and B-Spline curves

One of the limitations of B-Splines and Bézier curves is that it is not possible to represent conic sections and in their standard non-rational version, these curves can only approximate these forms. Rational curves overcome this limitation and offer one complete mathematical form for the precise representation of the standard analytical shapes. Rational forms have added flexibility in the form of weights which can be used to modify the curve.

A rational B-Spline curve of order $k$ is defined as

$$
\begin{equation*}
\mathbf{c}(u)=\frac{\sum_{i=1}^{n} w_{i} \mathbf{d}_{i} N_{i, k}(u)}{\sum_{i=1}^{n} w_{i} N_{i, k}(u)} \tag{4.6}
\end{equation*}
$$

where the basis functions $N_{i, k}(u)$ are the usual B-Spline basis functions of order $k$ defined on a knot set $\left\{u_{i}\right\}_{i=1}^{n+k}$.

Similarly, a rational Bézier curve of degree $m$ is defined as

$$
\begin{equation*}
\mathbf{c}(t)=\frac{\sum_{i=0}^{m} B_{i}^{m}(t) w_{i} \mathbf{b}_{i}}{\sum_{i=1}^{m} B_{i}^{m}(t) w_{i}} \quad 0 \leq t \leq 1 \tag{4.7}
\end{equation*}
$$

The $w_{i}$ in these expressions are known as weights of the rational B-Spline or rational Bézier curves.

It is convenient to represent rational B-splines as a projection of 4-D entities in the so called homogeneous coordinates. We represent a point in 3-D $\left(E^{3}\right)$ in terms of points in 4D $\left(E^{4}\right)$, where the point $\mathbf{P}^{h}=(h x, h y, h z, h)$ in $E^{4}$, when normalised as $(x, y, z, 1)$, represents the point $\mathbf{P}(x, y, z)$ in $E^{3}$. The normalisation can be interpreted as a perspective map with its centre at the origin of $E^{4}$ on the hyper-plane $h=1$ ( $h$ being the fourth coordinate component, called the homogeneous coordinate). If we let $H$ denote this map, then it is defined exactly by

$$
H\{(h x, h y, h z, h)\}= \begin{cases}\left(\frac{h x}{h}, \frac{h y}{h}, \frac{h z}{h}\right) & h \neq 0 \\ \text { point at infinity on the line from the origin } & \\ \text { through the } \operatorname{point}(x, y, z) & h=0\end{cases}
$$

Figure 4.2 shows an analogy for the representation of 2-D points using 3-D homogeneous space.

In terms of these 4-D points we define a polynomial (i.e non-rational) B-spline curve of order $k$ by the formula

$$
\mathbf{c}^{h}(u)=\sum_{i=1}^{n} \mathbf{P}_{i}^{h} N_{i, k}(u) .
$$

Here the $N_{i, k}(u)$ are the normal $k$ th order B-spline basis functions, and the $\mathbf{P}_{i}^{h}$ are the 4 D control points in homogeneous space. As with conventional B-splines there is also an associated knot vector $\left(u_{i}\right)_{i=1}^{n+k}$.

The curve $\mathbf{c}^{h}(u)$ forms a set of points in 4D homogeneous space. We obtain the 3D rational form of the curve, $\mathbf{c}(u)$, by projecting $\mathbf{c}^{h}(u)$ into 3-D. As stated above this is achieved by


Figure 4.2: Projection from homogeneous space to curve space
dividing the first three coordinates of each 4-D point by its homogeneous coordinate:

$$
\mathbf{P}^{h}=(h x, h y, h z, h) \longrightarrow(x, y, z, 1) .
$$

For our rational B-spline curve $\mathbf{c}^{h}(u)$ the homogeneous coordinate is

$$
\sum_{i=1}^{n} h_{i} N_{i, k}(u)
$$

and so the rational B-spline curve $\mathbf{c}(u)$ takes on the form

$$
\begin{equation*}
\mathbf{c}(u)=\frac{\sum_{i=1}^{n} h_{i} \mathbf{P}_{i} N_{i, k}(u)}{\sum_{i=1}^{n} h_{i} N_{i, k}(u)} \tag{4.8}
\end{equation*}
$$

Equation 4.8 represents a piecewise rational function at the distinct knots in the sequence

| Element | Symbol |
| :--- | :--- |
| Rational B-Spline curve | $\mathbf{c}(u)$ |
| Basis function | $N_{i, k}(u)$ |
| Order of the curve | $k$ |
| Control points | $\mathbf{d}_{i} \quad i=1, \ldots, n$ |
| No of control points | $n$ |
| Weights | $w_{i}$ |
| No of weights | $n$ |
| Knots | $u_{i} \quad i=1, \ldots, n+k$ |
| No of knots | $n+k$ |
| Rational B-spline in |  |
| homogeneous coordinates | $\mathbf{c}^{w}(u)$ |
| Control points in <br> homogeneous coordinates | $\mathbf{P}_{i}^{h}$ or $\mathbf{d}_{i}^{w}$ |

Table 4.3: Summary of the notation for rational B-spline curves
$\left(u_{i}\right)_{i=k}^{n+1}$. The $h_{i}$ are substituted by weights which are usually represented by $w_{i}$ and using $\mathbf{d}_{i}$ for the B-spline control points in place of the $\mathbf{P}_{i}$, we obtain our original definition, equation 4.6.

By using homogeneous coordinates, it is possible to use non-rational algorithms developed for non-rational Bézier and $B$-spline curves in the rational case as explained in the next section.

### 4.1.4 NURB Algorithms

The algorithms considered in this section are:

1. Degree elevation, which is used when we want to to represent a curve of a given degree as one of a higher degree. This procedure increases flexibility of a control polygon by providing more vertices but leaving the curve shape unchanged.
2. The de Casteljau algorithm, which is a special case of the B-spline recursion formula (de Boor algorithm) when applied to Bézier curves.
3. Subdivision, which allows to split a curve into parts that conserve the shape of the curve.
4. Evaluation algorithm.
5. Knot insertion algorithm.
6. Derivative evaluation.

We can apply the algorithms developed for non-rational B-spline curves and surfaces to the rational forms simply by applying the non-rational algorithms to 4D version of the entity in homogeneous space and then dividing through. In the curve case we start with 4D points $\left(w_{i} \mathbf{x}_{i} w_{i}\right)^{T}$, apply the algorithm to these points and obtain another set of 4 D points $\left(\mathbf{y}_{i} v_{i}\right)^{T}$. From these we obtain the required 3D points as $\mathbf{y}_{i} / v_{i}$. The rational weights of these 3D points are the numbers $v_{i}$. Effectively, we apply the algorithm to the non-rational 4D representation of the curve or surface and then project the result onto the plane $w=1$ in 4D, by dividing through by the fourth coordinate. For positive weights it is numerically more stable to divide through by the 4 th coordinate at each stage of the calculation so insuring that each intermediate control point lies in the convex hull of the original polygon.

## Degree elevation

It is often useful to be able to represent a curve of a given degree as one of a higher degree. It increases the flexibility of a control polygon by providing more vertices but leaving the curve shape unchanged. There are also important uses for degree elevation in surface construction. For example, we may wish to construct a surface interpolating to a series of cross section Bézier curves. This often leads to the requirement that all the curves be of the same degree. In this case degree elevation can be used to elevate all input curves to the one of highest degree.

For the non-rational case, if we are given a Bézier curve of degree $m$ with control points $\left(\mathbf{b}_{j}\right)_{j=0}^{m}$ and we wish to represent it as a Bézier curve of degree $m+1$ with vertices $\left(\mathbf{b}_{j}^{(1)}\right)_{j=0}^{m+1}$ say, then we require the following equation to hold:

$$
\sum_{j=0}^{m} \mathbf{b}_{j}\binom{m}{j} t^{j}(1-t)^{m-j}=\sum_{j=0}^{m+1} \mathbf{b}_{j}^{(1)}\binom{m+1}{j} t^{j}(1-t)^{m+1-j}
$$



Figure 4.3: Degree elevation process for a Bézier curve

If we multiply the left hand side by $(t+(1-t))=1$ we get

$$
\sum_{j=0}^{m} \mathbf{b}_{j}\binom{m}{j}\left(t^{j}(1-t)^{m+1-j}+t^{j+1}(1-t)^{m-j}\right)=\sum_{j=0}^{m+1} \mathbf{b}_{j}^{(1)}\binom{m+1}{j} t^{j}(1-t)^{m+1-j}
$$

Now we compare coefficients of $t^{j}(1-t)^{m+1-j}$ on both sides to obtain

$$
\begin{equation*}
\mathbf{b}_{j}^{(1)}=\frac{j}{m+1} \mathbf{b}_{j-1}+\left(1-\frac{j}{m+1}\right) \mathbf{b}_{j}, \quad j=0, \ldots, m+1 . \tag{4.9}
\end{equation*}
$$

Hence the new control points $\mathbf{b}_{j}^{(1)}$ are obtained from the old ones by piecewise linear interpolation at the parameter values $j /(m+1)$ (fig. 4.3). The new control polygon lies within the convex hull of the old one, i.e it is closer to the curve.

The process of degree elevation may be repeated, so allowing us to elevate the degree of a Bézier curve to any higher degree. After $r$ degree elevations the control polygon has vertices
$\mathbf{b}_{0}^{(r)}, \ldots, \mathbf{b}_{m+r}^{(r)}$, each $\mathbf{b}_{i}^{(r)}$ being given explicitly by the formula

$$
\mathbf{b}_{i}^{(r)}=\sum_{j=0}^{m} \mathbf{b}_{j}\binom{m}{j} \frac{\binom{r}{i-j}}{\binom{m+r}{i}},
$$

(where $\binom{r}{i-j}=0$ if $i-j<0$ or $i-j>r$ ).
Although repeated degree elevation will ensure that the control polygon eventually approaches the curve very closely (arbitrarily closely in the limit), the convergence is very slow (in contrast to subdivision convergence). Hence this property of degree elevation has no real practical applications.

Take the rational Bézier curve of degree $m$ :

$$
\mathbf{c}(t)=\frac{\sum_{i=0}^{m} w_{i} \mathbf{b}_{i} B_{i}^{m}(t)}{\sum_{i=0}^{m} w_{i} B_{i}^{m}(t)}
$$

We apply the degree elevation algorithm to the 4 D control points $\left(w_{i} \mathbf{b}_{i} w_{i}\right)$ and then divide through. This gives us 3D control points

$$
\mathbf{b}_{i}^{1}=\frac{w_{i-1} \alpha_{i} \mathbf{b}_{i-1}+w_{i}\left(1-\alpha_{i}\right) \mathbf{b}_{i}}{w_{i-1} \alpha_{i}+w_{i}\left(1-\alpha_{i}\right)}, i=0, \ldots, m+1
$$

with $\alpha_{i}=i /(m+1)$.
For a B-spline curve of the form

$$
\mathbf{c}_{k}^{w}(u)=\sum_{i=1}^{n} N_{i, k}(u) \mathbf{P}_{i}^{w}
$$

i.e. a $k$ th order (degree $p=k+1$ ) rational B-spline on the knot vector $U$, it is possible to elevate its degree to $p+1$ to the curve

$$
\mathbf{c}_{k+1}^{w}(u)=\mathbf{c}_{k}^{w}(u)=\sum_{i=1}^{\hat{n}} N_{i, k+1}(u) \mathbf{Q}_{i}^{w}
$$

over the knot vector $\hat{U}$ with control points $\mathbf{Q}_{i}^{w}$. Piegl and Tiller[52] present an algorithm for
the case when the knot set has the form

$$
U=\{\underbrace{a, a, \ldots, a}_{k}, u_{k+1}, \ldots, u_{n}, \underbrace{b, \ldots, b}_{k}\},
$$

which can be obtained by applying the knot insertion algorithm described below. The steps are:

1. Find the knot set $\hat{U}$ given by $\left(u_{i}\right)_{i=1}^{n+k+1+r}$, where $r$ is the number of segments making up $\mathbf{c}$ and the knots corresponding to the segment boundaries have their multiplicity increased by one.
2. Extract the $i$ th Bézier segment from the curve by knot insertion;
3. degree elevate the $i$ th Bézier segment;
4. remove unnecessary knots separating the $(i-1)$ th and $i$ th segments.

## de Casteljau algorithm

The de Casteljau recursion can be summarised with the formula:

$$
\begin{equation*}
\mathbf{b}_{i}^{r}(t)=(1-t) \mathbf{b}_{i}^{r-1}+t \mathbf{b}_{i+1}^{r-1} \quad 0 \leq r \leq m \tag{4.10}
\end{equation*}
$$

where $m$ is the degree of the Bézier curve and $\mathbf{b}_{i}^{0}$ are the original control points $\mathbf{b}_{i}$. An important property of the recursion is that

$$
\mathbf{b}(t)=\mathbf{b}_{0}^{m}(t)
$$

which makes the recursion a suitable method for the evaluation of Bézier curves. This is graphically represented for a cubic Bézier in figure 4.4.

A rational Bézier curve may be evaluated by applying the de Casteljau algorithm to the 4 D control polygon $\left(w_{i} \mathbf{b}_{i} w_{i}\right)$ and then dividing through, that is we apply the algorithm to


Figure 4.4: Graphical representation of the de Casteljau recursion
the Bézier curve

$$
\sum_{i=0}^{m} w_{i} \mathbf{b}_{i} B_{i}^{m}(t)
$$

and to

$$
\sum_{i=0}^{m} w_{i} B_{i}^{m}(t)
$$

and simply divide the two results to get the desired point.
Although this is simple and usually effective, as pointed out above, it is not guaranteed to be numerically stable. If some of the weights $w_{i}$ are large, the intermediate control points $w_{i}^{r} \mathbf{b}_{i}^{r}$, from the numerator calculation, are no longer in the convex hull of the original control polygon and this may result in a loss of accuracy.

A more expensive but more stable method is to process the 4 D non-rational version of the curve:

$$
\mathbf{c}(t)=\sum_{i=0}^{m} \mathbf{b}_{i}^{h} B_{i}^{m}(t)
$$

with control polygon $\mathbf{b}_{i}^{h}=\left(w_{i} \mathbf{b}_{i} w_{i}\right)^{T}$, and project every intermediate de Casteljau point $\left(w_{i}^{r} \mathbf{b}_{i}^{r} w_{i}^{r}\right)^{T}$ onto the plane $w=1$. This gives us the rational de Casteljau algorithm:

$$
\mathbf{b}_{i}^{r}(t)=(1-t) \frac{w_{i}^{r-1}}{w_{i}^{r}} \mathbf{b}_{i}^{r-1}+t \frac{w_{i+1}^{r-1}}{w_{i}^{r}} \mathbf{b}_{i+1}^{r-1},
$$

with

$$
w_{i}^{r}(t)=(1-t) w_{i}^{r-1}(t)+t w_{i+1}^{r-1} .
$$

Note that for positive weights, the $\mathbf{b}_{i}^{r}$ are all in the convex hull of the original control polygon $\mathbf{b}_{i}$, (i.e $(1-t)\left(w_{i}^{r-1} / w_{i}^{r}\right)+t\left(w_{i+1}^{r-1} / w_{i}^{r}\right)=1$ ) so assuring numerical stability.

## Subdivision

Although a Bézier curve is usually defined over $[0,1]$, it can also be defined over any interval $[0, c]$. The de Casteljau algorithm supplies both the control points for the part of the curve over $[0, c]$ and the control points for the part of the curve over $[c, 1]$. For the cubic case, if we display the triangular array of points obtained using the de Casteljau recursion

the control points for the Bézier curve over $[0, c]$ are the points on the leading diagonal, $\mathbf{b}_{0}^{i}, \quad i=0, \ldots, m$, and those for the interval $[\mathrm{c}, 1]$ are the points on the trailing diagonal, $\mathrm{b}_{i}^{m-i}, \quad i=0, \ldots, m$. The two resulting Bézier segments are

$$
\begin{gathered}
\mathbf{p}_{[0, c]}(t)=\sum_{i=0}^{m} \mathbf{b}_{0}^{i} B_{i}^{m}(t), \quad t \in[0,1], \\
\mathbf{p}_{[c, 1]}(t)=\sum_{i=0}^{m} \mathbf{b}_{i}^{m-i} B_{i}^{m}(t), \quad t \in[0,1] .
\end{gathered}
$$

As in the non-rational case we may use the de Casteljau algorithm to subdivide a rational Bézier curve. We use the de Casteljau algorithm to subdivide the 4D version of the curve.

The intermediate 4D points $\left(w_{i}^{r} \mathbf{b}_{i}^{r} w_{i}^{r}\right)^{T}$ are then projected onto the plane $w=1$ by dividing through by the fourth coordinate. This provides us with the control polygons for the left and right hand segments of the rational curve. The control points and weights corresponding to the curve over $[0, t]$ are given by

$$
\begin{array}{rll}
\mathbf{b}_{i}^{l e f t}=\mathbf{b}_{0}^{i}, \quad w_{i}^{l e f t}=w_{0}^{i} & \text { over }[0, t] \\
\mathbf{b}_{i}^{r i g h t}=\mathbf{b}_{i}^{m-i}(t), & w_{i}^{r i g h t}=w_{i}^{m-i} & \text { over }[t, 1]
\end{array}
$$

In the cubic case we generate the following triangular arrays:

$$
\begin{array}{llll}
\mathbf{b}_{0}^{0}(t) & & & \\
& \mathbf{b}_{0}^{1}(t) & & \\
\mathbf{b}_{1}^{0}(t) & & \mathbf{b}_{0}^{2}(t) & \\
& \mathbf{b}_{1}^{1}(t) & & \mathbf{b}_{0}^{3}(t) \\
\mathbf{b}_{2}^{0}(t) & & \mathbf{b}_{1}^{2}(t) & \\
& \mathbf{b}_{2}^{1}(t) & & \\
\mathbf{b}_{3}^{0}(t) & & &
\end{array}
$$

where

$$
\mathbf{b}_{i}^{r}(t)=(1-t) \frac{w_{i}^{r-1}}{w_{i}^{r}} \mathbf{b}_{i}^{r-1}+t \frac{w_{i+1}^{r-1}}{w_{i}^{r}} \mathbf{b}_{i+1}^{r-1},
$$

and

$$
\begin{array}{llll}
w_{0}^{0}(t) & & & \\
& w_{0}^{1}(t) & & \\
w_{1}^{0}(t) & & w_{0}^{2}(t) & \\
& w_{1}^{1}(t) & & w_{0}^{3}(t) \\
w_{2}^{0}(t) & & w_{1}^{2}(t) & \\
& w_{2}^{1}(t) & & \\
w_{3}^{0}(t) & & &
\end{array}
$$

where

$$
w_{i}^{r}(t)=(1-t) w_{i}^{r-1}(t)+t w_{i+1}^{r-1}(t) .
$$

## Rational B-spline evaluation

This algorithm follows along much the same lines as the de Casteljau one. For the nonrational B-spline interpolation algorithm with

$$
\mathbf{c}(u)=\sum_{i=1}^{n} \mathbf{d}_{i} N_{i, k}(u)
$$

on the knot set $\left(u_{i}\right)_{i=1}^{n+k}$, we find $i$ such that $u_{i} \leq u<u_{i+1}$ and then compute

$$
\mathbf{d}_{j}^{r}(u)=\alpha_{j}^{r} \mathbf{d}_{j}^{r-1}(u)+\left(1-\alpha_{j}^{r}\right) \mathbf{d}_{j-1}^{r-1}(u), \quad \mathbf{d}_{i}^{0}=\mathbf{d}_{i}
$$

where

$$
\alpha_{j}^{r}=\frac{u-u_{i}}{u_{i+k-r}-u_{i}},
$$

obtaining to the following triangular table

$$
\begin{array}{lllll}
\mathbf{d}_{i-k+1}^{0}(u) & & & & \\
\mathbf{d}_{i-k+2}^{0}(u) & \mathbf{d}_{i-k+2}^{1}(u) & & & \\
\vdots & & \ddots & & \\
\mathbf{d}_{i-1}^{0}(u) & \mathbf{d}_{i-1}^{1}(u) & \ldots & \mathbf{d}_{i-1}^{k-2}(u) & \\
\mathbf{d}_{i}^{0}(u) & \mathbf{d}_{i}^{1}(u) & \ldots & \mathbf{d}_{i}^{k-2} & \mathbf{d}_{i}^{k-1}(u)
\end{array}
$$

(so that the indices are $j=i-k+r+1, \ldots, i ; \quad r=0, \ldots, k-1$ ). The required point is then

$$
\mathbf{c}(u)=\mathbf{d}_{i}^{k-1}(u) .
$$

If the evaluation point is an already existing knot value with multiplicity $s$ say, we can
use a reduced table. The triangular array now takes on the reduced form

$$
\begin{array}{llll}
\mathbf{d}_{i-q}^{0}(u) & & & \\
\mathbf{d}_{i-q+1}^{0}(u) & \mathbf{d}_{i-q+1}^{1}(u) & & \\
\vdots & & \ddots & \\
\mathbf{d}_{i}^{0}(u) & \mathbf{d}_{i}^{1}(u) & \ldots & \mathbf{d}_{i}^{q}(u)
\end{array}
$$

(that is for $j=i-q+r, \ldots, i ; \quad r=0, \ldots, q$ ) with $q=k-1-s$. The required result is then

$$
\mathbf{c}(u)=\mathbf{d}_{i}^{q}(u)
$$

Returning to the rational case, we can either apply the above algorithm to the numerator and denominator of the B-spline curve:

$$
\mathbf{c}(u)=\frac{\sum_{i=1}^{n} w_{i} \mathbf{d}_{i} N_{i, k}(u)}{\sum_{i=1}^{n} w_{i} N_{i, k}(u)}
$$

and divide through (although again this can lead to instabilities), or, we apply the algorithm to the non-rational B-spline curve in homogeneous coordinates

$$
\mathbf{c}^{h}(u)=\sum_{i=1}^{n} \mathbf{d}_{i}^{h} N_{i, k}(u)
$$

with the 4D control points $\mathbf{d}_{i}^{h}=\left(w_{i} \mathbf{d}_{i} w_{i}\right)^{T}$, and project the intermediate points $\left(w_{i}^{r} \mathbf{d}_{i}^{r} w_{i}^{r}\right)^{T}$ onto the plane $w=1$ :

$$
\mathbf{d}_{j}^{r}(t)=\alpha_{j}^{r} \frac{w_{j}^{r-1}}{w_{j}^{r}} \mathbf{d}_{j}^{r-1}+\left(1-\alpha_{j}^{r}\right) \frac{w_{j-1}^{r-1}}{w_{j}^{r}} \mathbf{d}_{j-1}^{r-1}
$$

with

$$
w_{j}^{r}(t)=\alpha_{j}^{r} w_{j}^{r-1}(t)+\left(1-\alpha_{j}^{r}\right) w_{j-1}^{r-1} .
$$

## Knot insertion

Knot insertion carries through to the rational case in the same way as the above algorithms. To insert the knot $\hat{u}$ coinciding with the knot $u_{i+1}$ which has multiplicity $s$ say ( $s=0$ if it doesn't already appear), we apply one step of the above B-spline recursion algorithm

$$
\mathbf{d}_{j}^{1}=\alpha_{j}^{1} \mathbf{d}_{j}+\left(1-\alpha_{j}^{1}\right) \mathbf{d}_{j-1},
$$

where

$$
\alpha_{j}^{1}= \begin{cases}1 & j \leq i-k+s+1 \\ \left(\hat{u}-u_{j}\right) /\left(u_{i+k-1}-u_{i}\right) & i-k+s+2 \leq j \leq i \\ 0 & j \geq i+1\end{cases}
$$

so that the original B-spline control points $\left(\mathbf{d}_{j}\right)_{j=i-k+2+s}^{i-1}$ are replaced by the points $\left(\mathbf{d}_{j}^{1}\right)_{j=i-k+2+s}^{i}$.
For the rational algorithm we apply this method and calculate

$$
\left.\mathbf{d}_{j}^{1}(t)=\alpha_{j}^{1}\right) \frac{w_{j}}{w_{j}^{1}} \mathbf{d}_{j}+\left(1-\alpha_{j}^{1}\right) \frac{w_{j-1}}{w_{j}^{1}} \mathbf{d}_{j-1}
$$

with

$$
w_{j}^{1}(t)=\alpha_{j}^{1} w_{j}(t)+\left(1-\alpha_{j}^{1}\right) w_{j-1}(t) .
$$

This gives replacement control points $\left(\mathbf{d}_{j}^{1}\right)_{j=i-k+2+s}^{i}$ and replacement weights $\left(w_{j}^{1}\right)_{j=i-k+2+s}^{i}$.
Subdivision of rational B-spline curves follows along similar lines. Using the above algorithm we simply insert the knot corresponding to the splitting point until it has multiplicity $k-1$. The resulting control points and associated weights then split into two groups, one for the left hand rational B-spline curve and the other for the right hand part.

## Derivative evaluation

We can evaluate the derivatives of a rational Bézier curve as follows. Write

$$
\mathbf{c}(t)=\frac{\sum_{i=0}^{m} w_{i} \mathbf{b}_{i} B_{i}^{m}(t)}{\sum_{i=0}^{m} w_{i} B_{i}^{m}(t)}=\frac{\mathbf{p}(t)}{w(t)},
$$

where

$$
\mathbf{p}(t)=\sum_{i=0}^{m} w_{i} \mathbf{b}_{i} B_{i}^{m}(t), \quad w(t)=\sum_{i=0}^{m} w_{i} B_{i}^{m}(t)
$$

Then $\mathbf{p}(t)=w(t) \mathbf{c}(t)$ and

$$
\mathbf{p}^{\prime}(t)=w^{\prime}(t) \mathbf{c}(t)+w(t) \mathbf{c}^{\prime}(t)
$$

so

$$
\mathbf{c}^{\prime}(t)=\frac{1}{w(t)}\left[\mathbf{p}^{\prime}(t)-w^{\prime}(t) \mathbf{c}(t)\right] .
$$

For higher derivatives, we differentiate $\mathbf{p}(t) r$ times to get

$$
\mathbf{p}^{(r)}(t)=\sum_{j=0}^{r}\binom{r}{j} w^{(j)}(t) \mathbf{c}^{(r-j)}(t) .
$$

We then solve for $\mathbf{c}^{(r)}(t)$ :

$$
\mathbf{c}^{(r)}(t)=\frac{1}{w(t)}\left[\mathbf{p}^{(r)}(t)-\sum_{j=1}^{r}\binom{r}{j} w^{(j)}(t) \mathbf{c}^{(r-j)}(t)\right] .
$$

This is a recursive formula for the $r$ th derivative of a rational Bézier curve. Note that it only involves taking derivatives of polynomial curves. At the endpoints of the curve we have

$$
\begin{gathered}
\mathbf{c}^{\prime}(0)=\frac{m}{w_{0}}\left[w_{1} \mathbf{b}_{1}-w_{0} \mathbf{b}_{0}-\left(w_{1}-w_{0}\right) \mathbf{b}_{0}\right] . \\
=\frac{m w_{1}}{w_{0}}\left(\mathbf{b}_{1}-\mathbf{b}_{0}\right) .
\end{gathered}
$$

Similarly, we obtain

$$
\mathbf{c}^{\prime}(1)=\frac{m w_{m-1}}{w_{m}}\left(\mathbf{b}_{m}-\mathbf{b}_{m-1}\right) .
$$

As with non-rational forms the Bézier curve is tangent to the first and last legs of the control polygon.

Derivatives of a rational B-spline curve can (as in the non-rational case) be conveniently be computed using the above knot insertion algorithm. If we wish to evaluate the derivative at a point $\hat{u}$, we simply insert this knot until it has multiplicity $k-1$. The curve control points and weights then behave like the rational Bézier form around $\hat{u}$, and hence the derivatives can be computed by using the recursion formula for the derivative of a rational Bézier curve.

Note that for derivative evaluation one cannot just apply the corresponding non-rational B-spline algorithm to the numerator and denominator and divide. The quotient rule must be used. A rational B-spline curve has a rational Bézier representation. As in the nonrational case we can obtain the Bézier points and weights by inserting all knots until they have multiplicity $k-1$.

### 4.2 Surface Modelling

### 4.2.1 Tensor Product Bézier Surfaces

The idea of tensor product surfaces is to mix the creation of curves along perpendicular axes, as done in the creation of curves by sweeping a deforming curve (figure 4.5) along guides which are themselves curves.

The mathematical representation of this concept is as follows[16]: let the initial curve be a Bézier curve of degree $m$ :

$$
\mathbf{b}^{m}(u)=\sum_{i=0}^{m} \mathbf{b}_{i} B_{i}^{m}(u)
$$

Making each $\mathbf{b}_{i}$ move over a Bézier curve of degree $n$ :

$$
\mathbf{b}_{i}=\mathbf{b}_{i}(v)=\sum_{j=0}^{n} \mathbf{b}_{i, j} B_{j}^{n}(v)
$$



Figure 4.5: Creation of a surface by sweeping a deforming curve. This principle is the basis for the tensor product surface analogy

By combining these equations we obtain a Bézier surface patch $\mathbf{b}^{m, n}$

$$
\begin{equation*}
\mathbf{b}^{m, n}(u, v)=\sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{i, j} B_{i}^{M}(u) B_{j}^{n}(v) . \tag{4.11}
\end{equation*}
$$

The characteristic properties of a tensor product surface are:

- a double sum,
- a matrix of points $\left(\mathbf{b}_{i, j}\right)_{i, j=0}^{m, n}$ known as the control net for the surface and,
- a basis set $\left(B_{i}^{m}(s) B_{j}^{n}(t)\right)_{i, j=0}^{m, n}$

The general surface can be represented with a rational B-Spline surface or a rational Bézier surface. As in the case of curves, the two representations are analogous and algorithms have been developed to transform a representation from one form to the other.

### 4.2.2 B-Spline surfaces

Just as Bézier tensor product surfaces are generalisations of the curve formulation so B-spline surfaces are tensor product generalisations of B-spline curves. A B-spline surface of order $k$
by $l$ takes the form

$$
\mathbf{x}(u, v)=\sum_{i=1}^{p} \sum_{j=1}^{q} \mathbf{d}_{i j} N_{i, k}(u) N_{j, l}(v),
$$

where associated with $\mathbf{x}$ is a knot set in $u,\left(u_{i}\right)_{i=1}^{p+k}$ and a knot set in $v,\left(v_{j}\right)_{j=1}^{q+l}$. The array of points $\left(\mathbf{d}_{i j}\right)_{i, j=1}^{p, q}$ form the control net for the B-spline surface analogous to the Bézier case. The surface itself is defined over the interval $\left[u_{k}, u_{p+1}\right] *\left[v_{l}, v_{q+1}\right]$, the other knots in $u$ and $v$ being the extra ones added to form a basis set for the $u$ and $v$ directions. The basis functions in the surface case consist of the products of the curve basis functions:

$$
\left(N_{i, k}(u) N_{j, l}(v)\right)_{i, j=1}^{p, q}
$$

The surface $\mathbf{x}$ consists of as many patches in the $u$ direction as there are distinct internal knots in the knot sequence $\left(u_{i}\right)_{i=k}^{p+1}$, and for $v$ the sequence $\left(v_{j}\right)_{j=l}^{q+1}$. The product of these two numbers gives us the number of patches making up $\mathbf{x}$. Because the individual segments in $u$ and $v$ are defined locally by $k$ and $l$ basis functions respectively, altering a particular control point of the net will effect the surface only locally (fig. 6.3), changing at most $k l$ patches making up the B-spline surface. In particular, altering a control point $\mathbf{d}_{i j}$ of the surface affects $\mathbf{x}$ only in the range $\left[u_{i}, u_{i+k}\right) *\left[v_{j}, v_{j+l}\right)$, the range over which the associated basis function $N_{i, k}(u) N_{j, l}(v)$ is defined.

### 4.2.3 Rational B-Spline surfaces

Rational B-spline and Bézier surfaces are direct generalisations of the rational curves. We define a B-spline surface of order $k$ by $l$ in 4D homogeneous space as

$$
\mathbf{x}^{h}(u, v)=\sum_{i=1}^{p} \sum_{j=1}^{q} \mathbf{d}_{i j}^{h} N_{i, k}(u) N_{j, l}(v)
$$

where $\mathbf{d}_{i j}^{h}$ is the 4 D point

$$
\left(\begin{array}{llll}
d_{i j}^{x} w_{i j} & d_{i j}^{y} w_{i j} & d_{i j}^{z} w_{i j} & w_{i j}
\end{array}\right)
$$

Dividing through by the homogeneous coordinate

$$
\sum_{i=1}^{p} \sum_{j=1}^{q} w_{i j} N_{i, k}(u) N_{j, l}(v)
$$

we obtain the rational B-spline surface $\mathbf{x}(u, v)$ :

$$
\mathbf{x}(u, v)=\frac{\sum_{i=1}^{p} \sum_{j=1}^{q} \mathbf{d}_{i j} w_{i j} N_{i, k}(u) N_{j, l}(v)}{\sum_{i=1}^{p} \sum_{j=1}^{q} w_{i j} N_{i, k}(u) N_{j, l}(v)}
$$

As with non-rational B-spline surfaces there is an associated knot vector in $u,\left(u_{i}\right)_{i=1}^{p+k}$, and in $v,\left(v_{j}\right)_{j=1}^{q+l}$. The points $\mathbf{d}_{i j}$ form the rational control net for the surface and can be interpreted as the projection of the 4D non-rational control net formed from the $\mathbf{d}_{i j}^{h}$. By writing $\mathbf{x}(u, v)$ as

$$
\sum_{i=1}^{p} \sum_{j=1}^{q}\left(\frac{w_{i j} N_{i, k}(u) N_{j, l}(v)}{\sum_{i=1}^{p} \sum_{j=1}^{q} w_{i j} N_{i, k}(u) N_{j, l}(v)}\right) \mathbf{d}_{i j}
$$

we see that the rational B-spline basis functions are given by

$$
R_{i, k ; j, l}(u, v)=\frac{w_{i j} N_{i, k}(u) N_{j, l}(v)}{\sum_{r=1}^{p} \sum_{s=1}^{q} w_{r s} N_{r, k}(u) N_{s, l}(v)}
$$

For simplicity we write this as $R_{i, j}(u, v)$. In order for these basis functions to be non-negative we require the following conditions on the weights $w_{i j}$ :

$$
w_{11}, w_{p 1}, w_{1 q}, w_{p q}>0, \quad w_{i j} \geq 0 \text { otherwise. }
$$

Note that $R$ is not a product function, i.e. it is not the product of the rational curve basis functions, as was the case for non-rational B-spline surfaces. However, they are similar in shape to the standard non-rational basis functions, $N_{i, k}(u) N_{j, l}(v)$, and have analogous properties:

- $R_{i, j}(u, v) \geq 0$.
- $\sum_{i=1}^{p} \sum_{j=1}^{q} R_{i, j}(u, v) \equiv 1$.
- Local support: $R_{i, j}(u, v)=0$ if $(u, v)$ is outside the rectangle $\left[u_{i}, u_{i+k}\right) *\left[v_{j}, v_{j+l}\right)$. Furthermore, in any given rectangle $\left[u_{i}, u_{i+1}\right) *\left[v_{j}, v_{j+1}\right)$ at most $k l$ of the basis functions are non-zero.
- The $R_{i, j}(u, v)$ functions are generalisations of the $N_{i, k}(u) N_{j, l}(v)$ product B-spline surface basis functions. If we set all the weights to 1 we recover the non-rational form.

Because of the similarity of the basis functions to the non-rational ones, a rational Bspline surface enjoys all the important properties of the non-rational form, e.g convex-hull, local modification etc.

### 4.2.4 Rational Bézier patches

The rational Bézier surface patch of degree $m$ by $n$ takes on the form

$$
\mathbf{x}(s, t)=\frac{\sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{i j} w_{i j} B_{i}^{m}(s) B_{j}^{n}(t)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i j} B_{i}^{m}(s) B_{j}^{n}(t)}, \quad(s, t) \in[0,1] *[0,1] .
$$

The $\mathbf{b}_{i j}$ form the rational control net, the projection of the 4-D control net formed from the $\mathbf{b}_{i j}^{h}$. The basis functions are given by

$$
\frac{w_{i j} B_{i}^{m}(s) B_{j}^{n}(t)}{\sum_{p=0}^{m} \sum_{q=0}^{n} w_{p q} B_{p}^{m}(s) B_{q}^{n}(t)}
$$

A rational Bézier surface is a special case of a rational B-spline surface on the knot set $(0, \ldots, 0,1, \ldots, 1)$ in $u$, where 0 and 1 occur with multiplicity $m+1$, and $(0, \ldots, 0,1, \ldots, 1)$ in $v$, where the multiplicity is $n+1$. Setting all the weights to be equal recovers the nonrational Bézier surface patch. A composite rational Bézier surface can be considered as a
rational B-spline surface with the internal knots in $u$ occurring with multiplicity $m$ and those in $v$ with multiplicity $n$.

### 4.2.5 Surface algorithms

The surface algorithms, e.g degree elevation, knot insertion, subdivision, etc. follow the same pattern as the non-rational ones. We apply the non-rational tensor product algorithms to the 4-D version of the surface, consisting of control points $\left(\begin{array}{ll}w_{i j} \mathbf{d}_{i j} & w_{i j}\end{array}\right)$. Since the non-rational tensor product extensions amount to using just the curve algorithms we can utilise the stable form of the rational curve algorithms, where appropriate, by projecting each intermediate point of the calculation onto the plane $w=1$.

### 4.3 The IGES Standard

The Initial Graphics Exchange Specification, IGES is an ANSI standard for the transfer of graphics and geometry data. IGES was developed in the early 1980s and it is the most widely used format for data exchange among CAD/CAM/CAE systems.

IGES includes in the definition of the standard representations for curves, surfaces, three dimensional solids and finite element content. An example stripped IGES file is in figure 4.6.

### 4.3.1 IGES file structure

The IGES format was defined originally to contain only ASCII, human readable characters in a 80 character per record format. The binary form of the standard was defined later, as the file sizes increased. However, most implementations use the ASCII form.

An IGES file consists of five sections (figure 4.7):

1. Start section. This section is a region of readable text at the beginning of the file which is used for documentation.
2. Global section, which contains parameters such as file name, author, date of creation. precision of figures, etc.


Figure 4.6: Example of an IGES file

## GLOBAL (G)

> DIRECTORY ENTRY (D)

PARAMETER DATA (P)

TERMINATE (T)

Figure 4.7: Structure of an IGES file
3. Directory Entry (DE) section. This section contains an index to the parameter section, as a list of entities, along with various descriptive attributes (e.g. colour, line type). An entry in the DE section consists of two lines that hold 20 fields of eight characters each.
4. Parameter Data (PD) section. This section gives entity definitions, e.g.. control points, knot data, endpoints of a line, etc.
5. Termination section. This section is one record in length. It contains a total number of records in each of the other sections.

The binary format adds a sixth section which contains binary formatted data.
All data is described in terms of entities. There are entities available to describe curves, surfaces, solids and so on. Table 4.4 shows some of the available entities in the standard. Some entity types are further subdivided by form numbers, resulting in a larger number of different entities.

### 4.3.2 Example specification: a NURBS curve and a NURBS surface

Entities 126 and 128 of the IGES specification describe a NURBS curve and a NURBS surface respectively.

A NURBS curve is specified in IGES by:

Entity Description Curves
116 Point
110 Line
100 Circular arc
104 Conic arc
112 Parametric spline curve
126 Rational B-Spline curve
102 Composite curve
Surfaces
118 Ruled surface
120 Surface of revolution
122 Tabulated cylinder
108 Plane
114 Parametric spline surface
128 Rational B-Spline surface
Constructive Solid Geometry
150 Block
158 Sphere
160 Torus
168 Ellipsoid
180 Boolean tree
184 Solid assembly
B-Rep Solid
186 Manifold solid B-Rep object
502 Vertex
504 Edge
508 loop
510 Face
514 Shell
Other entities
124 Transformation matrix
106 Copious data
134 Node - FEA geometric point
136 Element - FEA element topology
138 Nodal displacement / rotation
148 Load/constraint - FEA non-geometric content

Table 4.4: Some entities in the IGES specification

| (1) 128 | $(2)$ Param. <br> Data | (3) <br> Struct. | (4) <br> Line Font Pattern | (5) Level | (6) | (7) | (8) <br> Label <br> Display | (9) | (10) <br> Sequence <br> Number <br> D \# |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| (11) 128 | (12) <br> Line <br> Weight | (13) <br> Color <br> Number | (14) <br> Param. <br> Line <br> Count | (15) <br> Form <br> Number | (16) | (17) | (18) <br> Entity <br> Label | (19) | (20) <br> Sequence <br> Number <br> D \# + 1 |

Records in the DE section:

| 128 | 309 |  | 1 | 75 | 0 | 0 | OD |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: |
| 128 | 2 | 6 | 26 | 0 |  |  | OD |
| 237 |  |  |  |  |  |  |  |

Records in the P section:

| $128,3,5,3,3,0,0,1,0,0,0.0,0.0,0.0,0.0,1.0,1.0,1.0,1.0,0.0,0.0$, | 237 P | 309 |
| :--- | :--- | :--- |
| $0.0,0.0,0.375,0.625,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0$, | 237 P | 310 |
| $1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0$, | 237 P | 311 |
| $1.0,-9.59884107409246,251.87999,11.9501753952658$, | 237 P | 312 |
| $-3.22039928938901,251.87999,3.35432175325268,3.66603838737259$, | 237 P | 313 |
| $251.87999,-4.84080572029901,10.900289418313,251.87999$, | 237 P | 314 |
| $-12.7495769867665,-9.91405061952486,244.476799233967$, | 237 P | 315 |
| $12.0742953564063,-3.3727440728984,244.476799233967$, | 237 P | 316 |
| $3.57368038714364,3.75315162422275,244.476799233967$, | 237 P | 317 |
| $-4.444671925977,11.2910799913377,244.476799233967$, | 237 P | 318 |
| $-12.0882234103884,-10.2721379300126,232.129566754316$, | 237 P | 319 |
| $12.3372782126316,-3.49551336607463,232.129566754317$, | 237 P | 320 |
| $3.95586862047848,3.90220343933955,232.129566754316$, | 237 P | 321 |
| $-3.87554048902915,11.785322674788,232.129566754316$, | 237 P | 322 |
| $-11.2280852695849,-11.5366474860555,212.39786526405$, | 237 P | 323 |
| $11.4396390835815,-3.7523788346899,212.39786526405$, | 237 P | 324 |
| $3.90904870495048,4.67933278440071,212.39786526405$, | 237 P | 325 |
| $-2.89087526630634,13.5346389204137,212.39786526405$, | 237 P | 326 |
| $-9.1481196011557,-12.5479896297906,200.088748772983$, | 237 P | 327 |
| $10.3847615280325,-4.06920901593887,200.088748772983$, | 237 P | 328 |
| $3.69206345061693,5.063495425367,200.088748772983$, | 237 P | 329 |
| $-2.10218797901534,14.700625305664,200.088748772983$, | 237 P | 330 |
| $-7.03181043114952,-13.1117393022491,192.70129,9.78440277594289$, | 237 P | 331 |
| $-4.21818217002092,192.70129,3.56354424967022,5.33149029986496$, | 237 P | 332 |
| $192.70129,-1.67358162401593,15.3856272774944,192.70129$, | 237 P | 333 |
| $-5.84892969895996,0.0,1.0,0.0,1.0,0,0 ;$ | 237 P | 334 |

Figure 4.8: Directory section table and example of B-Spline surface definition in IGES

- its degree $p$;
- the number of control points $n$;
- Euclidean control points, $\mathbf{d}_{i}$;
- weights $w_{i}$, which have to be positive;
- its knot vector, $U$, containing $m+1=n+p+1$ knots;
- start and end parameter values, $s_{0}$ and $s_{1}$;
- other nonessential but useful information, e.g. whether the curve is planar or nonplanar, open or closed, truly rational ( $w_{i}$ not all equal), etc.

The IGES specification has no concept of homogeneous control points, $\mathbf{P}_{i}^{h}$. The formula given for a rational B-spline curve in the IGES specification is:

$$
\mathbf{G}(t)=\frac{\sum_{i=0}^{K} W_{i} \mathbf{P}_{i} b_{i}(t)}{\sum_{i=0}^{K} W_{i} b_{i}(t)}
$$

A NURBS surface is defined analogously by:

- the degrees $p$ and $q$ in each parameter direction $u$ and $v$;
- the number of control points $n$ and $m$;
- Euclidean control points $\mathbf{P}_{i, j}$;
- weights $w_{i, j}$;
- knot vectors for each parameter direction $\left\{u_{i}\right\}_{i=1}^{r=n+p+1}$ and $\{v\}_{j=1}^{s=m+q+1}$, which fulfill the condition $u_{i-1} \leq u_{i}$ and $v_{j-1} \leq v_{j}$ for $i=2, \ldots, r$ and $j=2, \ldots, s$.

The parameters $s_{0}, s_{1}$ and $t_{0}, t_{1}$ define the intended surface. The surface can be tagged as special, e.g. a plane, circular cylinder, cone, sphere, torus. The formula for the rational
$B$-spline surface in the specification is:

$$
\mathbf{G}(s, t)=\frac{\sum_{i=0}^{K_{1}} \sum_{j=0}^{K_{2}} w_{i j} \mathbf{P}_{i j} b_{i}(s) b_{j}(t)}{\sum_{i=0} K_{1} \sum_{j=0}^{K_{2}} w_{i j} b_{i}(s) b_{j}(t)}
$$

Figure 4.8 shows an example of a description of a NURBS surface in an IGES file. Two lines describe the entity in the directory entry section and 26 lines in the parameter section of the IGES file give the parameters, which are: The parameters listed are:

1. Entity type number (128),
2. $K_{1}(3) ; K_{1}+1$ is the number of control points in the direction $s$ (4), integer;
3. $K_{2}(5) ; K_{2}+1$ is the number of control points in the direction $t(6)$, integer;
4. $M_{1}$, degree of first set of basis functions (3), integer;
5. $M_{2}$, degree of second set of basis functions (3), integer;
6. PROP1, $0=$ Not closed in first parametric variable direction, integer;
7. PROP2, $0=$ Not closed in second parametric variable direction, integer;
8. $\operatorname{PROP} 3,0=$ Rational/ $1=$ Polynomial (1), integer;
9. PROP4, $0=$ Non-periodic in first parametric variable direction $/ 1=$ Periodic (0), integer;
10. PROP5, $0=$ Non-periodic in second parametric variable direction $/ 1=$ Periodic ( 0 ). integer;
11. $2+K_{1}+M_{1}(8)$ values of the first knot sequence,
12. $2+K_{2}+M_{2}$ (10) values of the second knot sequence,
13. $\left(1+K_{1}\right) *\left(1+K_{2}\right)(24)$ values of the weights,
14. the control point coordinates $\left(24^{*} 3\right)$ in their three components and finally
15. the starting and ending values of the parameters in both directions.

Using the IGES specification, it is possible to communicate to a graphics engine or a CAGD application to display a custom made NURBS surface.

### 4.4 Solid Modelling representations

Solid modelling is concerned with representations that are "complete" and are thus suitable for any geometric queries to be solved algorithmically[38].

There are three major approaches to represent solid models:

- Boundary models (Boundary Representation Models or B-Rep).
- Decomposition models.
- Constructive models (Constructive Solid geometry or CSG).


### 4.4.1 Boundary Representation Models

These method of representation represent a point set in terms of its boundary. The boundary is usually a collection of faces. Faces may be again represented again by their boundaries, which are lines or one-dimensional curves. Because of this decomposition, the model may be viewed as a hierarchy of models[38, p.56]. Unfortunately, an arbitrary set of non-overlapping faces does not necessarily correspond to the boundary of a solid. Early versions of many solid modellers were plagued with invalid B-Reps due to designer faults or incorrect algorithms[55].

### 4.4.2 Constructive Solid Geometry Models

These models represent a point set as a combination of primitive point sets. Each of the primitives is represented as an instance of a primitive solid type (e.g. a block, a cylinder, etc.) Constructive models include operations such as boolean operations, which are more general construction operations.

CSG is the most popular constructive representation. The primitives may be simple shapes or complex features for particular applications.

The closure of operations in the r-set space is guaranteed by the use of the regularization operation (closure of interior or $\operatorname{clo(int()))\text {.Theregularizationalwaysreturnsvalid(although}}$ possibly empty) solids.

### 4.4.3 Decomposition models and voxel modelling

Solids may be represented by a variety of space decomposition schemes. The entire 3-D space, or just the set that corresponds to the solid, is partitioned into non-overlapping 3-D regions called cells. The most usual type of cells used is the voxel, which refers to a volume cell. Each voxel is a rectangular cuboid with six faces, twelve edges and eight corners. An alternative definition for a voxel from the previous one is to identify the voxel with the actual sample of a volumetric variable over a structured rectilinear grid (see figure 4.9).


Pixels


Voxels

Figure 4.9: Pixels and voxels

A solid is represented by a collection of cells from a fixed collection of primitive cell types, combined with a single "gluing" operation. Regular decompositions may have a significant error because of the discretised representation, but they are nevertheless popular because the simplicity of the scheme is well-suited to parallel algorithms and hardware support.

### 4.5 The voxel model

Although the voxel model is presented in this context as a method to represent geometry, the representation of geometry is a subset of the representations possible with the technique. A voxel model is a special case of a general volumetric data set, which typically is a set $S$ of samples $(x, y, z, v)$, representing the value $v$ of some property at a certain location $(x, y, z)$. The samples may be taken at random locations in space, but in many cases $S$ is isotropic. containing samples taken at regularly spaced intervals along three orthogonal axes. For a geometric representation as described above, it suffices for $v$ to represent either true or false, to represent that the region is either part of the solid or not. The value $v$ may contain, however, more information than a binary digit, such as integers, vectors or higher order entities.

Voxel models are used in medical imaging (e.g. CT, MRI), biology (e.g. con-focal microscopy), geo-science (e.g. seismic measurements, oil exploration), industry (e.g. nondestructive inspection) and chemistry (e.g. electron density maps)[36, 29].

### 4.5.1 Memory requirements for the voxel model

Voxel modelling is a poor representation scheme when it comes to conciseness. A voxel model requires huge amounts of memory. To achieve a good resolution (e.g. $5 \mu \mathrm{~m}$ ) in a considerable volume (e.g. $500 \mathrm{~mm} \times 500 \mathrm{~mm} \times 500 \mathrm{~mm}$ ) requires $100,000^{3}$ voxels, i.e. $10^{15}$ elements. To grasp the enormity of this figure, according to some rough estimates, the information of all U.S. academic libraries together is twice that amount, roughly $2 \times 10^{15}$ bytes. (see table 4.5). These memory requirements are beyond current computer system's capabilities. Modelling at this scale seems unfeasible unless there is a breakthrough in computing technology. Current high-end systems (1998) can handle a $1024 \times 1024 \times 1024$ element data set using hardware optimised for 3-D graphics.

The storage requirement of a voxel model is

$$
n \times n \times n \times \sum_{l=1}^{n_{p}}\left(p_{l}\right)
$$

Table 4.5: Estimates of the quantities of data contained by the various media (adapted from [62])

| Memory unit | Size in bytes | Example of media |
| :---: | :---: | :---: |
| Byte | $2^{0}=1$ | A single character |
| Kilobyte | $2^{10} \approx 1,000$ | A very short story |
| Megabyte | $2^{20} \approx 1,000,000$ | A small novel |
| Gigabyte | $2^{30} \approx 1,000,000,000$ | Ten meters of shelved books |
| Terabyte | $2^{40} \approx 10^{12}$ | $1 / 2$ of an academic research library |
| Petabyte | $2^{50} \approx 10^{15}$ | $1 / 2$ of all US academic research libraries |
| Exabyte | $2^{60} \approx 10^{18}$ | - |
| Zettabyte | $2^{70} \approx 10^{21}$ | - |
| Yottabyte | $2^{80} \approx 10^{24}$ | - |

where $n_{p}$ is the number of properties and $p_{l}$ is the storage requirement of a value of the property $l$. Typical voxel models in medicine are based on a value of $n=512$, and store a single density property represented by an integer. In this case, the voxel model occupancy is around 512 MB . In other application areas, such as in earth sciences, the memory storage could be increased by 10 to 50 times. This is the major drawback of voxel models [5].

### 4.5.2 Compression Methods

In principle, the compression methods in the 3-D domain are a generalisation of the compression methods available in 2-D for working with raster images. As with their 2-D counterparts, there are lossless and lossy compression methods.

A short list of methods includes,

- Compression based on the DCT (Discrete Cosine Transform) [65].
- Compression based on wavelets $[48,49,50]$.
- Fractal compression [12].
- Multi-resolution representations [10, 11].
- Compression based on hierarchical structures:

Octree and BSP trees [42].

Many of these techniques have been studied in relationship with their application for medical imaging and volume rendering. In fact medical imaging equipment often uses either the raw voxel model or a octree model for the visualisation [60].

The octree representation uses a recursive subdivision of the space of interest into eight octants that are arranged into an 8 -ary tree (hence the name). This type of structure is analogous to the quadtree which is used in 2-D raster image processing. The octant volumes continue to be subdivided until a termination criterion is satisfied. Two common termination criterion are the total volume represented by a node and the complexity (homogeneity) of the volume represented by the node.

The representation of a solid object by exhaustive enumeration, without regard to its material composition, requires a binary value: either a voxel is internal to the solid or it is external to the solid. In this case, the octree can compress the volume of data by aggregating large regions where this binary value is either zero or one.

The classical octree models this using three types of nodes: white, black and grey. The octree divides the space into cubes which are inside or outside the object. Node types are defined in the usual way.

- White: The corresponding octant is homogeneous and external to the solid.
- Black: The corresponding octant is homogeneous and internal to the solid.
- Grey: The corresponding octant is heterogeneous, i.e. parts of it are internal to the solid.

In general, the number of nodes in this type of octree representation of a solid object is proportional to the surface area of the object. Hence octree models are not quite as large as exhaustive representations but still take a fair amount of storage [38]. This scheme for the classical octree can be built from a geometrical model of a solid (boundary, CSG or voxel-based representation).

To build an octree representation from a volume voxel model the procedure is different. since the volume voxel model does not necessarily represent a solid. Voxel models are used


## OBJECT REPRESENTATION



Figure 4.10: Octree subdivision scheme. A three level representation of an object
to represent fog and clouds, which are amorphous in nature. These models have been used to render realistic scenes including fog and cloud models.

A isosurface octree, or the classical octree of a voxel mode defines voxels as black if their associated value is within a specific range of the property and white otherwise. The voxels whose property values are within this range and differ less than a given $\varepsilon$ are recursively grouped into black nodes. This type of octree is only useful when the volume is not very heterogeneous [5].

In the case of medical imaging, for example, it is not enough to store a given isosurface, since it is important to conserve the information of the volume. Several researchers have used the octree data structure, or a variation of it, to reduce the data access time [60]. In this type of application, each node of the octree contains a value that corresponds to the average value of the associated property across the octant volume represented by the node. The root node of the tree represents the entire object space volume, and leaf nodes correspond to volumes that are homogeneous, or nearly so. Leaf nodes do not represent identically sized volumes; instead they represent object space volumes that satisfy the termination criteria. For the homogeneity criterion, leaves or leaf nodes represent volumes having the same value of the associated property.

The extended octree or vector octree are based on storing a boundary representation in the nodes of an octree. The vector octree stores boundaries of a polyhedral object within the cells of an octree. The octant subdivision is continued until each cell contains at most one vertex, one edge, one face, or is homogeneously "full" or "empty" [38, 5]. These structures have been developed to model solid homogeneous objects.

### 4.5.3 Manufacturing FGMs from a voxel model

Considering the fabrication of FGMs through powder stacking or spray deposition, the use of a voxel model seems practical. It is reported that for powder stacking, powders of sizes from $15 \mu \mathrm{~m}$ to $44 \mu \mathrm{~m}$ are used, and that layer-by-layer stacking of powders allows controlling the spatial distribution to 0.2 mm , while spray deposition allows control to a minimum size of $0.01 \mathrm{~mm}[45$, p.165]. The material to be used in the laser fusion project at De Montfort

University is H 10 tool steel powder with a distribution of sizes from $63 \mu \mathrm{~m}$ to $125 \mu \mathrm{~m}[61]$.


Figure 4.11: Scanning Electron Microscopy image of H10 tool steel powder[61]

These figures mean that small and medium sized objects of up to about $200 \times 200 \times 200$ $\mathrm{mm}^{3}$ could be modelled with voxels using current computer technology, assuming a model of $1000 \times 1000 \times 1000$ elements and the stated resolution of $0.2 \mathrm{~mm}(200 \mu \mathrm{~m})$. This resolution produces a rough surface. Modelling at the resolution required for a smooth surface finish (around $5 \mu \mathrm{~m}$ ) involves heavy requirements of memory and computing power.

## Chapter 5

## Modelling multi material and FGM

## objects

Kumar et al.[33] go through a short review of various approaches to modelling heterogeneous objects, including their earlier work [35, 34]. Several solid model representations are mentioned for geometric domain representation:

- Manifold solids,
- R-sets,
- S-sets,
- Selective geometric complexes (SGCs),
- Non-manifold solids and
- Constructive non-regularized geometry (CNRG).

Going beyond the geometric representation domain, the methods listed are:

- Heterogeneous solid models,
- Chain models,
- Hermite hyperpatches and
- FR-sets.

The paper proceeds then to present their proposed object models.
Another review of possible modelling methods is presented in [47].

### 5.1 Work at the University of Michigan

The following subsections discuss the methods proposed for the representation for heterogeneous objects $[34,35]$ and a more general object model [33].

### 5.1.1 Representation of heterogeneous objects

To be able to represent multiple materials, a material dimension M is added to the spatial dimensions $\mathbf{R}^{\mathbf{3}}$ that capture the geometry and topology of an object. For a finite number of unique materials, the choice for the material dimension $\mathbf{M}$ would be the set of integers $\mathbf{I}$. Then the product space $\mathbf{T}=\mathbf{R}^{\mathbf{3}} \times \mathbf{I}$ with the product topology can form a new modelling space for representing multiple-material objects.

A solid described using traditional solid modelling techniques is a member of the class of r-sets $\mathbf{A}$ in $\mathbf{R}^{\mathbf{3}}$. The method proposes a new class $\mathbf{A}_{\mathbf{m}}=\mathbf{A} \times \mathbf{K}$, where $\mathbf{A}$ is the class of r-sets and $\mathbf{K} \subset \mathbf{I}$ is a finite set of integers. Each material is characterised by an integer in $\mathbf{K}$. A typical member $Q \in \mathbf{A}_{\mathbf{m}}(Q=\{P, k\})$ is called an $r_{m}$-set and is composed of an r-set $P \in \mathbf{A}$ and an integer $k \in \mathbf{K}$

This definition is extended to represent functionally graded materials. To model objects with continuous material variation, the material space must be expanded from $\mathbf{K} \subset \mathbf{I}$ in the previous case. A suitable choice for the new mathematical space is $\mathbf{T}=\mathbf{R}^{\mathbf{3}} \times \mathbf{R}^{\mathbf{n}}, n$ being the number of primary materials. $\mathbf{R}^{3}$ is the geometry space, where geometry and topology are defined, using a traditional solid model (CSG, B-Rep or hybrid). $\mathbf{R}^{\mathbf{n}}$ is the material space. The material can be identified at any point by volume fractions of each of the primary materials. Since the volume fractions must sum one unit, the space of volume fractions is a
subspace $\mathbf{V} \subset \mathbf{R}^{\mathbf{n}}$, such that

$$
\mathbf{V}=\left\{\mathbf{v} \in \mathbf{R}^{\mathbf{n}} /\|\mathbf{v}\|_{1} \equiv \sum_{i=1}^{n} v_{i}=1 \wedge v_{i} \geq 0\right\}
$$

where $v_{i}$ represents the volume fraction of material $i$.
Each point in an object $S$ can now be characterised in product space $\mathbf{T}$ as ( $\mathbf{x}, \mathbf{v}(\mathbf{x}))$ where $\mathbf{x} \in S$ is a point in the object and $\mathbf{v}(\mathbf{x}) \in \mathbf{V}$ represents the material at that point. The geometry of an object $S$ can be modelled as an r-set $P$ and the material distribution for the r-set $P$ can be represented by the set $B \in \mathbf{V}$ which is defined by a function $\mathbf{F}$, mapping the geometric points $\mathbf{x}$ to the material space $\mathbf{V}$. Hence, the representation for the object is:
$S=(P \in \mathbf{A}, B \subseteq \mathbf{V})$ where $B=\{\mathbf{v}(\mathbf{x}) \equiv \mathbf{F}(\mathbf{x}) \in \mathbf{V}, \forall \mathbf{x} \in P\}$
The authors also define modelling operations on $r_{m}$-sets and on $r_{m}$-objects. The representation allows for the usual set operations, such as difference, intersection, and union. A new operation called join is defined which combines two $r_{m}$-sets into a single $r_{m}$-set if the two material functions corresponding to each $r_{m}$-set are identical and can be combined into a single $C^{\infty}$ function. The modelling operations include the $\oplus$ operation that operates on the material components of the $r_{m}$-sets. For example, when two $r_{m}$-sets are intersected, the material properties of the resulting $r_{m}$-set are defined using the $\oplus$ operation. A trivial type of $\oplus$ operator would take, for instance, the material properties of either of the objects. A more elaborate $\oplus$ operator would try to combine the volumetric fractions of each of the components. The type of operator and the representation of the result are not discussed further.

Figure 5.1 shows the computer representation proposed for an $r_{m}$-object. The structure is a combination derived from the data structure of the commercial ACIS modeller with an additional material related data structure (shown on the right) added to the original structure (shown on the left). The form of the MFUNC block of the diagram is crucial to the representation method. Bhashyam et al.[3] have an actual implementation of this method (discussed below). The MFUNC element may also be modelled using a voxel-based approach. which is promising as it may be both versatile and free of the memory requirements which


Figure 5.1: Computer representation of an $r_{m}$-object (Kumar and Dutta[35])
make pure voxel-based modelling unattractive.
This work is extensive and mathematically rigorous. There are however blanks left, notably the treatment of the $\oplus$ operator, whose forms need to be explored to fully understand the algebraic properties of the operations on the objects represented.

An actual object was modelled and built on the Sanders Model-maker using this representation technique. The probe of smoothly varying volumetric fraction was built by modifying the tool path generation strategy. Given a certain layer distribution, there is currently no method for the automated generation of an optimal tool path for its fabrication.

### 5.1.2 Implementation of the $\mathbf{r}_{m}$-object representation

In [3], an actual implementation of the $\mathrm{r}_{m}$-object representation is presented. The 'Heterogeneous Solid Modeler' is a prototype CAD system based on the ACIS kernel (Spatial Technologies Inc) and whose GUI is implemented using Motif and OpenGL libraries. The architecture of the system is schematically presented in figure 5.2.


Figure 5.2: Architecture of the Heterogeneous Solid Modeler (Bhashyam et al.)

For the implementation of the heterogeneous material component, the authors suggest eight material composition functions, each suited to a particular application and specific geometries (e.g. box, cylinder, sphere, cone, torus, etc.). These would correspond to the MFUNC component in the computer representation (see figure 5.1). As previously noted, this function could also be implemented using a voxel model.

Additionally, the authors implement a set of property estimation methods to evaluate various thermo-mechanical properties. A general design cycle with the tool would include (see figure 5.3):

- Selection of the geometry.
- Selection of the materials, which are usually two, although the GUI presents options for up to four materials. The materials can be chosen form the ones available in an internal database, that includes the material properties.
- Input of the material composition function.
- Represent as a heterogeneous solid model ( $\mathrm{r}_{m}$-set)
- Repeat until all the primitives are modelled.
- Combine the heterogeneous primitives.
- Convert to a FE input file.
- Perform the FE Analysis and evaluate results
- If the results are not satisfactory, modify the composition function variables and repeat the conversion and analysis steps until the results are satisfactory.

The authors kindly supplied a copy of the prototype implementation on request, which was tested on a SPARC Ultra-250 using SunOS 5.7.

It is possible to input more than two materials in a primitive and this implementation seems to support up to four materials. The GUI provides a text box for the input of a material composition function through a formula. This is a very important step and it was
not possible to verify whether the input was correct. It is not clear from the context what type of expression is expected as a formula.

### 5.1.3 The object model

The object model[33] is a more general representation based on the concepts of product manifolds and trivial fibre bundles. The model is generic enough for many types of varying attributes of an object in a rigorous and integrated way. The authors recognise several characteristics of an object that require modelling: geometry, material, material properties and physical parameters. The base attribute and the most fundamental is the geometry. Each point in the object is described as one point in the Euclidean space $\mathbf{E}^{3}$.

The mathematical model $\mathcal{M}$ is the combination of several models, one model $\mathcal{M}_{G}$ for the geometry and one model $\mathcal{M}_{A_{i}}$ for each attribute $A_{i}$, but all based on the geometrical model.

$$
\mathcal{M}=\mathcal{M}_{G} \otimes \mathcal{M}_{A_{1}} \otimes \mathcal{M}_{A_{2}} \otimes \ldots \otimes \mathcal{M}_{A_{n}}
$$

The geometrical model is defined as

$$
\mathcal{M}_{G}=\left(P,\left\{C_{i}\right\}\right)
$$

where $P$ is an $r$-set in $\mathbf{E}^{\mathbf{3}}$ and $\left\{C_{i}\right\}$ is a finite set of disjoint decompositions of $P$. A decomposition $\left\{C_{i}\right\}$ consists of several 3-cells and forms a geometric cell complex. Each 3-cell $U_{\alpha}$ in $C_{i}$ possesses a local coordinate system, related to the global coordinate system through a coordinate map $\psi_{\alpha}$. Further constraints are imposed on each of these coordinate maps regarding the compatibility (non-vanishing Jacobian). A 3-cell $U_{\alpha}$ and its corresponding coordinate map $\psi_{\alpha}$ are called a chart and the collection of charts $C_{i}$ is called an atlas.

The attribute model is defined as

$$
\mathcal{M}_{A}=(N, F)
$$

The generic model for an attribute $A$ is a manifold $N$, which could be a vector or a tensor
space. Each point $\mathbf{x}$ in the r -set $P$ is mapped to its corresponding attribute through an attribute function $F . F$ is defined in a particular atlas $C_{j}$ and it can be subdivided for every chart $U_{\alpha}$ in the atlas into several mapping functions $F_{\alpha}$, where

$$
F_{\alpha}:\left(U_{\alpha} \in C_{j}\right) \rightarrow\left(V_{\gamma} \subseteq N\right)
$$

As indicated above, the object model $\mathcal{M}$ combines the geometry model $\mathcal{M}_{G}$ and the attribute models $\mathcal{M}_{A_{i}}$. If the model has a single attribute, the object is modelled in the space $S=P \times N$. For an object having $n$ attributes, the product set would be:

$$
S=P\left(\prod_{i=1}^{n} N_{i}\right)
$$

where $P$ is the r-set model describing geometry and each $N_{i}$ is a manifold describing the attribute $A_{i}$

The representation of an $\mathrm{r}_{m}$-object is a subset or a particular case of the object model described with only the material composition as an attribute.

### 5.2 Work at the MIT

Jackson et al. [24] present a method to represent materials with multiple materials and gradation of the materials based on the cell-tuple structure.

For the proposed representation, a model $M$ of a solid is subdivided in cells. Cells can have various geometries, although the examples presented show only tetrahedrons. It is mentioned that the tetrahedrons may also have curved faces, although the examples presented use planar faces. To represent the composition, the approach is the same as the one suggested by Kumar and Dutta, i.e. for every point $\mathbf{x} \in M$ use a vector valued function $\mathbf{m}(\mathbf{x})$ with components $m_{i}$ for every material $i$ present in the object. $m_{i}$ represents the volumetric fraction of a material in the object.

Hence, for a homogeneous cell $c_{\kappa}, \mathbf{m}=\mathbf{m}_{\kappa}, \forall \mathbf{x} \in c_{\kappa}$. For a heterogeneous cell, the object is defined by a set of control points $\left\{\mathbf{x}_{\kappa, i} / \mathbf{i} \mid=n_{g}\right\}$ and a set of control compositions

$$
\left\{\mathbf{m}_{\kappa, i} /|\mathbf{i}|=n_{m}\right\}
$$

These control points are combined with the barycentric Berstein polynomials:

$$
\left[\mathbf{x}_{\kappa}(\mathbf{u}), \mathbf{m}_{\kappa}(\mathbf{u})\right]=\left[\begin{array}{lll}
\sum_{|\mathbf{i}|=n_{g}} B_{i}^{n_{g}}(\mathbf{u}) \mathbf{x}_{\kappa, i} & , \quad \sum_{|\mathbf{i}|=n_{m}} B_{i}^{n_{m}}(\mathbf{u}) \mathbf{m}_{\kappa, i}
\end{array}\right]
$$

where $n_{g}$ and $n_{m}$ are the degrees of variation in shape and composition, $|\mathbf{i}|=i_{0}+i_{1}+\ldots+i_{k}$ and $k$ is the dimension of cell $\kappa$. $B_{i}^{n}(\mathbf{u})$ represents the ith Bernstein polynomial of degree $n$ and it is defined as:

$$
B_{i}^{n}(\mathbf{u})=\left[\frac{(n!)}{\left(i_{0}!\right)\left(i_{1}!\right) \ldots\left(i_{k}\right)}\right]\left[\left(u_{0}^{i_{0}}\right)\left(u_{1}^{i_{1}}\right) \ldots\left(u_{k}^{i_{k}}\right)\right]
$$

$\mathbf{u}=\left[u_{0}, u_{1}, \ldots, u_{k}\right]$ are the barycentric coordinates of a point in the domain and satisfies the condition

$$
|\mathbf{u}|=u_{0}+u_{1}+\cdots+u_{k}=1
$$

|i| represents an index composed of the sequence of values $i_{0}$ to $i_{k}$, e.g. 1000 or 1210 where every $i$ value is smaller or equal to $n_{g}$ but the condition $|\mathbf{i}|=n_{g}$ means that the the digits sum up to $n_{g}$

### 5.3 Discussion of the methods

These two methods are the response to the general lack of other methods to represent materially graded objects. The work at Michigan is a superset of the second method. The representation using B-splines is more straightforward and several algorithms available in computational geometry can be transfered to this application. Unfortunately, the geometric intuitiveness of B-Splines is lost in this representation which is no longer geometric.

To intimately work with the ACIS modelling kernel, the approach chosen by Dutta et al. at Michigan is practical. Many algorithms and test can be tested and applied without reinventing existing and proven CAD libraries for the heterogeneous case.


Figure 5.3: Design cycle using the heterogeneous solid modeller (Bhashyam et al.)

## Chapter 6

## Visualisation of voxel models

"A picture is worth a thousand words" goes the saying. For a human an image can convey a lot of information. For large datasets it is impractical and sometimes pointless to look at individual values or the raw data of the dataset. Through visual aids it is usually possible to understand the data in one look: it is the most natural way for humans to understand large amounts of information.

Visualisation is the technique used to explore scientific data through transformations and mappings. By mapping on a computer display visualisation uses computer graphics techniques such as rendering. Based on the broad definition though, some other tasks, such as contouring, classifying and generating physical 3-D models from scientific data through Rapid Prototyping can be regarded as visualisation too[46].

### 6.1 Exploration of a voxel model

A typical voxel model is a large number of values that cannot be grasped except through visualisation tools. This data can usually be interpreted as a series of images or 'slices' of data, as in the case of medical imaging voxel models. The voxel model is then the 3-D analogous of an image and the imaging methods can be extrapolated to the 3-D domain to explore the information.

Some tools to explore the data are: obtaining a histogram of values, slicing and cutting,
thresholding, resampling, contouring and volume rendering. All these are usual imaging techniques that are useful in 3-D too. Only volume rendering stands out from the rest as a uniquely 3 -D technique.

Medical imaging voxel models have been typically examined by watching individual images, each of which is a 'slice' of data in the whole voxel model. There are several applications available for these tasks, because medical imaging techniques have been around for several years.

Jackson et al. [24] refer to the use of voxel processing software for the creation of manufacturing models. By combining Rapid Prototyping in the visualisation tool-box, it has been possible to fabricate the model of a patient's skull directly from a patient's scan (figure 6.1). For example Mimics (Materialise n.v., Belgium) is an specialist medical imaging application capable of generating a surface model from a voxel model. The method used is the creation of a contour surface from a voxel model (see figure 6.2). This is explained in section 9.2.3.

Mimics can produce 3-D output in the industry standard STL format, VRML 2.0, IGES and a few other vendor-specific formats (e.g. Stratasys Layer Interface Files SSL). Materialise also offers the service of creating a physical 3-D model from medical scanner data in one week.

The most powerful method to display a voxel model is volume rendering, which has been used increasingly. Volume rendering is a computationally expensive method and software for volume rendering cannot display a model interactively, in real-time, while a user manipulates the point of view and camera positions.

An interactive exploration of voxel models with these tools is usually feasible with a powerful computer and enough memory. Some methods are computationally expensive and the performance degrades when using larger images (voxel models). Volume rendering is the most computationally expensive method and it is not interactive.

### 6.2 Volume Rendering

Volume Rendering is a method used to capture visually 3-D data sets in a 2-D image directly from 3-D volumetric data. Volume rendering differs from traditional computer graphics. which simulate a scene by rendering surfaces of a model. It also differs from image processing


Medical scanner data

Figure 6.1: Transformation of medical data to create a physical 3-D model (Materialise N.V.)


Figure 6.2: Surface contour created from a voxel model using VTK


Figure 6.3: Volume rendering operations
in that although the process may require image treatment, it is performed with a 3-D data set directly.

It is through volume rendering that spectacular see-through representations of human tissues are created, generating extremely informative medical images. That's a practical example of a suitable representation of a voxel model for a human to understand a model through vision.

Volume rendering is achieved by a sequence of operations in a "pipeline". The idea of a series of operations on a pipeline has been applied in visualisation software and it is a practical means of creating custom-built representations and generate visual representations of many different data sets.

The typical pipeline for volume rendering a 3-D model consists of: segmentation, gradient computation, resampling, classification, shading and compositing (see figure 6.3).

The order of operations can vary among implementations, for example classification can come before or after resampling.

Segmentation is a preprocessing step and it is typically done before the actual rendering. It is the process of separating the input data set into structural units, and something that needs to be done only once to the data set. Segmentation is a very difficult process and it is hard to capture it into an algorithm a computer can perform. Therefore, segmentation often requires the intervention of a human.

The gradient is a measure of how quickly voxel intensities in a data set change. The gradient indicates the direction of the change and how sharp the change is. Gradient computation is a computer intensive operation because the algorithm must traverse the entire data set, which is usually large. The gradient is used for the shading operations discussed below. The gradient can be computed using several operators, which approximate the continuous case using the discrete data available. The most commonly used gradient operators is the central difference gradient estimator, which uses six cells around the voxel to compute the gradient. One may choose to do the rendering using gradient operators which use all 26 cells surrounding a voxel. This is a computationally more expensive approach, but it may pay off as a key to visual understanding of the image.

An example of using different operators is shown in figure 6.4. The central difference gradient operator produces a smoother image, while the intermediate difference gradient operator registers the small holes on the left side of the skull behind the eye socket which are not visible using the central difference gradient operator.

To extract an image from the values in the volume data set, it is necessary to assign additional visual information to the voxels in the model.

- Colour: A colour can be assigned to a particular scalar voxel value through the use of a colour look-up table. The look-up table uses the voxel scalar value as input and assigns colours in the appropriate colour model. RGB is the most common colour model.
- Opacity: A value between zero and one, to indicate how opaque a voxel is. The opacity and the transparency are complementary, so that when the opacity is 1 , the transparency is 0 and vice versa.


Figure 6.4: The effect of gradient operators: central difference gradient operator (left) vs. intermediate difference gradient operator (right). (Lichtenbelt et al.[36])

These two tasks are usually done in the classification stage, which is done by an algorithm programmed in the rendering system. These values may also be calculated or assigned using the gradient computed earlier in the pipeline. It may be possible to associate certain segmentation information to the opacity or colour values. This could be used to make transparent a certain range of values which are not of interest. In medical imaging, for example, this could be used to make certain tissues transparent.

The resampling or interpolation stage is necessary, because the voxels in the data set and the pixels in the rendered image will seldom be aligned, and a ray cast to determine the rendered image will have to use values which are not in the original voxel space. During the resampling, new values are generated at new positions in voxel space.

Shading is used to render the image from the data set by using an illumination model. The illumination model describes the way a colour is assigned to a point in space, based on the light that shines on it, the angle between the viewer and the light, the material properties and the orientation and position in space. This calculation requires a surface normal for the reflection colour calculations. Since there is no surface in a volume model, the gradient operator is used to determine the angles for the calculations.

The last stage of the volume rendering pipeline, the compositing stage calculates the final colour of a ray. This operation, also called blending, calculates the colours of the final display image based on voxel transparency and colour.

### 6.3 Visualisation software

There are several visualisation applications available that can handled volume rendering of voxel models, among several other scientific visualisation tasks. There are many for medical imaging applications while some other (e.g. IRAF from the National Optical Astronomy Observatory) are application specific. Some of the available applications are[43, 51, 44]:

- VolVis
- GVLware(BoB)
- Application Visualization System (AVS)
- IBM Data Explorer (now OpenDX Open Visualization Data Explorer)
- IRIS Explorer
- Khoros
- PV-WAVE
- VoxelView
- Vis5D and VisAD
- Analyze
- Visualization Toolkit (VTK)

VolVis, developed at the State University of New York at Stony Brook, is a comprehensive volume visualisation system available in several (UNIX) platforms and as C source code. There have been several articles presented by the developers of this system [1.2,66].

The GVLware also known as BoB (Brick of Bytes) is a public domain application developed by the Army High Performance Computing Research Center (AHPCRC). This application is available for the Silicon Graphics platform.

AVS is a product from AVS Inc. AVS was the first large-scale, commercial visualisation system, dating back to 1989.[57, p.130] It is a mature product that went through several versions. The company was recently bought by Muse Technologies (May 2000).

IBM Data Explorer is a withdrawn product from IBM Inc. In May 1999, the IBM Open Visualization Data Explorer was announced to replace it. In their own words: "Open Visualization Data Explorer is a full visualisation environment that gives users the ability to apply advanced visualisation and analysis techniques to their data. These techniques can be applied to help users gain new insights into data from applications in a wide variety of fields including science, engineering, medicine and business. Data Explorer provides a full set of tools for manipulating, transforming, processing, realizing, rendering and animating data and allow for visualisation and analysis methods based on points, lines, areas, volumes, images or geometric primitives in any combination. Data Explorer is discipline-independent and easily adapts to new applications and data. The integrated object-oriented graphical user interface is intuitive to learn and easy to use." [22].

IRIS Explorer is a commercial product initially from Silicon Graphics Inc., bundled with Silicon Graphics workstations. It is also available from NAG Ltd. on other workstations and is supported by them. IRIS Explorer is a powerful visual programming environment for 3-D data visualisation, animation and manipulation. It is available on a broad range of PC and workstation platforms. OpenGL, Open Inventor and MasterSuite are some of the building blocks upon which IRIS Explorer is built. Like VTK, which is described below, Explorer works on the principle of constructing visualisation pipelines.

Khoros is a commercial product from Khoral Research Inc. It includes an integrated software development environment that allows users to compose and perform a variety of tasks related to image and signal processing, medical imaging, remote sensing, data exploration and scientific visualisation. Khoros includes a visual programming language, a suite of software development tools that extend the visual language and help you create new applications, an


Figure 6.5: Iris Explorer display of a visualisation pipeline
interactive user interface editor, an interactive image display package, 2-D/3-D plotting, and an extensive suite of image processing, data manipulation, scientific visualisation, geometry and matrix operators[51]. -WAVE is a commercial product from Visual Numerics Inc. It is a general purpose package, providing rich functionality suited to a wide range of users, although more particularly those with programming experience.

VoxelView, a product of Vital Images is used for medical imaging applications and is available solely for the Silicon Graphics platform.

VTK is not an application. It is a toolkit that application developers can use to create end-user products. VTK was used and extended in this project. VTK is described in the next section.

### 6.4 The Visualization Toolkit - VTK

The Visualization ToolKit (VTK)[57, 64, 30, 37] is an open source, freely available software system (a toolkit) for 3-D computer graphics, image processing, and visualisation. VTK includes a textbook (reference [57]), a C++ class library, and several interpreted interface layers including Tcl/Tk, Python, and Java. VTK has been implemented on nearly every Unix-based platform and PC's (Windows NT and Windows95). The design and implementation of the library has been strongly influenced by object-oriented principles.

The definitive guide to the toolkit is the textbook that comes with it, or the 'VTK Book' by Schroeder, Martin and Lorensen (reference [57]). This book focuses on the philosophy and design choices embedded in the toolkit, presents the algorithms and methods used. The 'VTK User's Guide' is aimed more at using the toolkit.

VTK visualisations work on the principle of constructing visualisation pipelines (figure 6.6). The idea of a visualisation pipeline is assembled using various building blocks to create specialised functions. VTK comes with dozens of components: various filters, sources, data readers and data generators. Additionally the developer can create custom-built components and add them to the library. A typical visualisation requires from 3 to 10 VTK classes[37].

The users of VTK range from students to application software engineers and researchers. Application developers can write applications in C++ and embed visualisation tasks in the


Figure 6.6: Representation of a visualisation pipeline
applications. By using the interpreted interface to the toolkit in either $\mathrm{Tcl} / \mathrm{Tk}$ or python, it is possible to produce prototype and try visualisation pipelines interactively. The possibility to access the toolkit through interpreted languages facilitates the creation and testing of prototype visualisation pipelines through interrogation and dialogue with the system in a scripting language (e.g. Tcl/Tk).

There is also a mailing list which is a forum used for user support and discussion of developments or bugs.

### 6.5 Design of the classes in the Visualization Toolkit

VTK was designed with its roots in animation and visualisation systems. It took 4 professionals 10 months to design the system with 25 classes which still sit at the centre of the software system, even after large extensions to its current size. This system was designed using Rumbaugh's OMT methodology developed at GE.

The design goals included as usual that the system should be robust, understandable, extensible, maintainable and reusable.

VTK is a large software project designed from the start with extensibility in mind. It consists of 514 classes and over 270,000 lines of code as of version 2.3 , released mid 1999. Lorensen reports about 600 classes by mid 2000 .

The classes are distributed in 5 groups or kits:


Figure 6.7: Visualisation of a combustor


Figure 6.8: Visualisation of a height profile


Figure 6.9: Distribution of the 270,000 lines of code among the kits in VTK (Adapted from Lorensen[37])

1. Common contains classes that re used by each of the other kits. These include abstract filter classes, datasets, cells and utility classes.
2. Graphics contains visualisation classes. These process vtk's dataset classes and render the resulting polygonal output. The visualisation filters in this kit extract surfaces of constant value, generate streamlines, warp surfaces and resample one data set with points from another.
3. Imaging contains classes that process volumetric image data, i.e. data with implied topology that is stored uniformly. Because of the uniform storage, these classes can be streamed and threaded.
4. Patented contains classes that implement techniques covered by US Patents. These classes can be used for educational purposes, but require a licence for commercial use
5. Contrib contains classes contributed by the vtk user community.


Figure 6.10: Distribution of the 514 classes among the kits in VTK (Adapted from Lorensen[37])

## Chapter 7

## Modelling FGMs

### 7.1 Element discretisation approach

A tool of choice for analysis of engineering objects is the Finite Element Method (FEM). This method is based on the subdivision of a whole part to be analysed through finite elements, and this approach can be used to model FGM objects. The element discretisation approach used for modelling FGM objects subdivides space into small cells. The material properties of a cell are assigned according to the material distribution. This approach has been used by König[31] in the optimisation of material composition using genetic algorithms and Kumar[32] in the optimisation of the material composition by minimising the compliance of a structure (maximising the rigidity).

### 7.2 Transforming a voxel model into a FEM grid

To represent a material as a list of cells with material properties assigned discretely to each is not the most effective way to represent an object in terms of the memory it requires. The representation of nodes and cells explicitly takes a large amount of memory, in contrast with the implicit structure of voxels. A FEM system could be implemented to apply algorithms on a structured points array (voxels). The translation from a voxel model to a cell model, an unstructured grid used in the finite element calculations is not unique. A simple translation
algorithm to create tetrahedral cells from a voxel model is presented in section 9.4. This algorithm creates six tetrahedral cells for every voxel.

The step of converting a voxel model into a mesh (unstructured grid) is required in order to use the FEM method on a standard tool. The use of a standard tool, such as ANSYS has the advantage of a gentler learning curve, built in graphics and mesh checking as well as company support and a user community.

However, a purpose-built FEM analysis program would still require the same number of nodes and it would still require to assemble a system of linear equations of the order of the number of degrees of freedom assigned in the model.

### 7.3 Estimation of the properties

The simulation of the behaviour of FGMs depends on the proper estimation of the properties for the interlayers. Miyamoto et al. recognise that "the most significant difficulty in FGM modeling is the accurate determination of the material properties of the interlayers" [45, p.64].

A simple approach for estimating the material properties of FGMs is the use of the rule of mixtures[45, p.68]. The most simple estimate is the classical linear rule of mixtures (Voight estimate) for two constituent materials:

$$
P=V_{\alpha} P_{\alpha}+V_{\beta} P_{\beta}
$$

where $P$ is a typical property and $V$ is the volume fraction. The subscripts $\alpha$ and $\beta$ are used to distinguish the two constituents. Another simple estimate is the harmonic mean (Reuss estimate):

$$
P=\frac{P_{\alpha} P_{\beta}}{V_{\alpha} P_{\beta}+V_{\beta} P_{\alpha}}
$$

These relationships are often used because of their simplicity. However, also because of their simplicity, their validity is limited.

For the examples presented here, the properties were evaluated using the Voigt estimate.

### 7.4 Examples of modelling technique

A corner bracket was modelled with the geometry and boundary conditions shown in figures 7.1 and 7.2


Figure 7.1: Geometry of the corner bracket modelled

The steps followed in this example were (see figure 7.3):

- Creation of a model of one single material (steel).
- Calculation of the stresses when submitted to the loading conditions.
- Based on the stress distribution obtained, re-assign the material of each element in the model. The range of average stresses was subdivided into five bins and an element's material was chosen according to which bin the element ended up in. The result was a model of five material mixtures.


Figure 7.2: Boundary conditions for the model

- Re-calculation of the stresses for the same loading conditions but now using the model of five material mixtures.

The results of this example are included in Appendix C.
A general method in an optimisation cycle may have similar character to the steps in the example. The key operations in the optimisation are material re-assignment and check of the end conditions.

The corner bracket example used 123 elements and 440 nodes. The element size was approximately $10 \mathrm{~mm} \times 10 \mathrm{~mm}$. The degrees of freedom of the model are approximately two per node, i.e. approximately 880 degrees of freedom.

Table 7.1: Element sizes and corresponding node counts for the corner bracket example

| Element size | Number of nodes | Number of degrees of freedom (approx.) |
| :---: | :---: | :---: |
| 10 mm | 440 | $\sim 880$ |
| $\sim 1.173 \mathrm{~mm}$ | 32,000 | $\sim 64,000$ |
| 1 mm | 44,000 | $\sim 88,000$ |
| $\sim 0.586 \mathrm{~mm}$ | 128,000 | $\sim 256,000$ |
| 0.1 mm | 4.4 million | $\sim 8.8$ million |

### 7.5 Discussion

The corner bracket example tests the possibility of modelling a material gradation using finite elements. The method is suited for the representation of multiple materials, because it is trivial to assign different attributes to different elements in the model.

The main difficulty of the technique is the scalability. This example uses a very coarse element size ( 10 mm ) and a small number of discrete steps in the material mixtures. The number of nodes required for an element size of about 1 mm would increase by a factor of 100 to 44,000 . The current ANSYS licence at our site limits the number of nodes to 32,000 (ANSYS/Multiphysics University High option), and the maximum number of nodes available with the software is 128,000 (ANSYS/Multiphysics Research Faculty option), which would permit element sizes of around 0.586 mm .


Figure 7.3: Operations performed in the corner bracket model


Figure 7.4: Optimisation of material distribution

## Chapter 8

## Software development

For the current project the software use revolved around tools used for visualisation of a voxel model and this was done using the Visualization Toolkit (VTK). The graphical notation and the key concepts used in this section correspond to the OMT methodology (Rumbaugh et al.[28]), whose concepts and notation are briefly presented in appendix B.

In this chapter I briefly explain what the framework of the VTK object model is, how it is used to represent data used for various visualisation tasks and how it was extended to store a voxel model as an octree.

### 8.1 VTK framework elements

A visualisation system deals with the data representation and its transformations. VTK proposes the concepts of the datasets and the pipelines to address each of these. "From an object-oriented viewpoint, transformations are processes in the functional model, while representations ate the objects in the object model"[57, p.84].

Schroeder et al. explain that object oriented purists may object to their design choices, which specifically separates operations from data objects.The reason for the 'unconventional' choice are the disadvantages of combining data and operations when the operations are much more complex than the data that they operate on, duplicating complex algorithms for several data types and the user's perception of how the operations are performed on the data. which
means that processes are naturally viewed as objects.
However, the system implements a few operations within data objects. The operations implemented within data objects were identified based on the authors' experience implementing visualisation algorithms.

### 8.1.1 The VTK visualisation pipeline

A visualisation process is concerned with transformations and mappings of data. To represent and abstract them, VTK proposes the idea of thinking of them as pipeline elements that can be assembled together. In this abstraction there are three types of elements: sources, filters and sinks.

- Source objects are for example data readers and geometry generators.
- Sink objects are data writers and mappers that ultimately will present their output on a computer display.
- Among these a pipeline usually has filters that perform the transformations from one data representation to another or combine several inputs.

The connections between pipes are checked for data type, because certain filters expect a specific type of input. Some filters require several inputs and may fan out to one or more branches of the visualisation pipeline. Sources and sinks usually have only one connexion: an output or an input, respectively.

The pipelines are often arranged along a line, which represent a series of operations to be performed in sequence. A pipe may also branch out, e.g. to show multiple representations of the same data, or several branches may merge in a filter or a mapper. The idea of pipelines doesn't usually lend itself to form loops unless there is control of when the execution of the operations in the pipeline occur.

The execution of operations can be either explicitly controlled by an 'executive' or can be implicitly requested every time that a part of the output requires output. Iris Explorer. AVS and IBM Data Explorer use the explicit approach, which lends itself better for parallel processing and distributed computing. VTK uses the implicit execution method, which is


1. A parameter modified
2. Executive performs dependency analysis
3. Executive executes necessary modules in order A-B-D-E

4. A parameter modified
5. E Output requested
6. Chain E-D-B-A back propagates Update() method
7. Chain A-B-D-E executes via Execute() method
(b) Implicit
(a) Explicit

Figure 8.1: Explicit vs. implicit network execution
simpler and modular, because a network element doesn't need to know about other objects except its inputs.

There are three proposed methods to interface with one's own data, depending on the complexity and sophistication required:

- Programming interface: Data is directly read, processed and written by a user's application. Data can be programmed and described through a program and fed into a network for visualisation. This is the most flexible approach, but requires the highest expertise.
- File interface: Data is prepared in a standard recognisable format and processed by using readers and writers implemented within VTK.
- System interface: This refers to the possibility of interfacing to other visualisation systems that manage whole scenes, with lighting, actors, cameras, geometries and transformations through exporters and importers. For example it is possible to use importers and exporters with VRML scenes, 3D Studio models and RenderMan RIB files.


Figure 8.2: A VTK dataset

### 8.1.2 Data representation in VTK

There are several desirable characteristics for the data representation in a visualisation system. The criteria for the design of VTK's data representation includes:

Compactness, to minimise memory requirements
Efficiency, to be able to access data easily, independently of the size of the data.
Mappability, to be able to represent visually the information without recourse to complex conversion processes.

Minimal coverage, which refers to achieving a minimal number of data types to represent efficiently visualisation data.

Simplicity, which is preferable in computational applications in order to understand and optimise the designs.

The data objects in the visualisation pipeline are called datasets. The object model diagram of this part of the toolkit is shown on figure 8.2.

The dataset has two parts: its geometry and its topology. The topology is represented through cells and instantiated to a specific geometry through points. There are 12 types of cells defined in VTK: Vertex, Polyvertex, Line, Polyline, Triangle, Triangle strip. Quadrilat-


Figure 8.3: Simple VTK cell types
eral, Pixel, Polygon, Tetrahedron, Hexahedron and Voxel. Figure 8.3 shows some of the cell types.

Data attributes can be associated either to the cells or the points, and there is a separate data attributes object for the data associated to the cells and the data associated to the points.

The data attributes object can contain, hold or be associated with several data types simultaneously. This is represented in figure 8.2 as the fan of associations spanning to the right of the vtkDataSetAttributes class. A vtkDataAttributes object does not store the attributes directly, but through relations with objects of several types: scalars, vectors, tensors, normals, texture coordinates and field data (the most general type).

Classes that store the information of several data types are specialisations from the attribute data class: vtkAttributeDataClass. There is one for every data type supported: vtkVectors, vtkTensors, vtkTCoords, vtkScalars, vtkNormals. The more general vtkFieldData, is a separate class altogether.

There are several specific types of datasets:

Polygonal data. This is data in the form of vertices, polyvertices, lines. polylines, triangles and triangle strips. This is unstructured data in one and two dimensions. This is a very useful representation for visualisation of surfaces in space. There is specialised hardware
that operates particularly on polygonal data with optimised speed. Particularly triangle strips are convenient to represent a surface through triangles with an efficient use of memory.

Structured Points. These are collections of points and cells arranged on a regular rectangular lattice. The points and cells are regularly arranged parallel to the global $x-y-z$ coordinates. The information required to represent the geometry and topology is minimal: only the coordinates of the origin, dimension of the dataset and the spacing are required. If only two dimensions are used, the dataset is referred to as a pixmap, a bitmap or an image.

Rectilinear Grid. This dataset has regular topology, aligned with the coordinate axes, but with irregularities in the spacing.

Unstructured Points. There is no topology associated with this dataset, and points are just a cloud in space with no structure associated to them. Vertices and polyvertices are used to represent unstructured points.

Unstructured Grid. This dataset is the most general one and it can hold any type of cell, unlike the polygonal data which is limited to 2-D primitives. Both the topology and the geometry are completely unstructured.

VTK's dataset hierarchy, shown in figure 8.5, shows the types of datasets available within the framework.

This data model may sometimes differ from the way a user may have his or her data, since the designers aim at representing most, but not all possible types of data. In rare cases the user may need to adapt the uses to the characteristics of the system.

Schroeder et al. discuss some other data models used in AVS (the Application Visualization System) and the Data Explorer (Haber, Lucas and Collins' model). In general they reckon that VTK's data model is not as abstract as that of either AVS or Haber's. The trade off in abstraction against simplicity was an intended design choice to make the system casier to use for the casual visualisation user.


Figure 8.4: Dataset types


Figure 8.5: Dataset object diagram

### 8.2 An extension to the toolkit

The structured points dataset was extended by a specialised class that stores the data in an spatial manner with the purpose of saving memory.

The vtkOctree class is derived from the vtkStructuredPoints class which comes in the original toolkit (see figure 8.6). The methods and support classes to achieve the storage of voxel information spatially are implemented and a small change in the vtkScalars is made to allow a subclass to override a method needed for this representation.

The data structure or data object created this way is then compared for memory usage with a few representative voxel models to quantify the memory savings.

The toolkit includes an object similar in nature but with a different purpose. The vtkPointLocator and the vtkCellLocator also implement an octree storage scheme but using a flat memory model and not using pointers. The purpose of the locators is to quickly locate cells and points in space. The difference is that for the same resolution or level, the octree implemented in the point locator uses more memory than a corresponding structured points dataset.


Figure 8.6: The vtkOctree class in VTK's object hierarchy

### 8.2.1 The octree dataset

A voxel mode is represented in VTK using vtkStructuredPoints. vtkOctree was derived from vtkStructuredPoints, because the data stored in an instance of this class is inherently the same as the data stored in the voxel model, only the method of storage changes from a regular storage to spatially referenced storage.

The classes involved in the implementation are:

- vtkOctree
- vtkOctreeNode
- vtkOctreeScalars

The object diagram in figure 8.7 shows the relationship among the three classes.
The associations among the three classes aligned at the top of the figure, vtkOctree, vtkPointData and vtkScalars are the same associations represented in figure 8.2, omitting for clarity the association with vtkCellData and showing the specialised classes vtkOctree and vtkPointData instead of vtkDataSet and vtkDataSetAttributes, again for clarity. In this case these associations are inherited from the respective superclasses.

A new association is required in this diagram to link cyclically the vtkOctreeScalars to a vtkOctree. The messages passed between the objects vtkOctree and vtkOctreeScalars


Figure 8.7: Object model for the octree
are used to map the structured coordinates and id as used in a standard structured points dataset and the mapping done in vtkOctreeScalars.

The id in a GetScalarid message passed to an object of class vtkStructuredPoints refers to the structured coordinates as described in figure 8.8. This id mapped through an octree that stores spatially the ids of scalars stored in the vtkOctreeScalars (see figure 8.9)

## vtkOctree

This class implements the data set concept. It is derived from vtkStructuredPoints.
The member functions are:

AddScalar implements the storage of values in an associated vtkOctreeScalars object. It returns the scalarId associated with the value stored, after verifying the homogeneity condition.

Initialize creates the octree spatial structure from an input dataset in the form of structured points.

ComputePointId(ijk[3]) returns the pointId at a position $i-j-k$ given in structured coordinates.


Figure 8.8: The structured coordinates of a vtkStructuredPoints dataset

## vtkOctreeScalars



Figure 8.9: Mapping of scalarId to values with vtkOctreeScalars

ArrayValue3D (vtkStructuredPoints *sp, int $x$, int $y$, int $z$ ), and auxiliary function that is used in the creation of the octree.

In this implementation, the homogeneity was chosen to be that the values are equal. This is practicable as the only values in the voxel model treated are values that indicate the material, for example $0,1,2$. In a different situation, the homogeneity condition could implement checks of the value being added to be 'close' to a stored value, e.g.

$$
|n e w-o l d|<t o l
$$

where new is the new value to be added, old is a value already present in the octree and tol is a given tolerance to determine the granularity of the representation.

PowerOf2Dimension is used to store a value of the form $2^{\text {level }}$, i.e. a power of 2 , so that $2^{\text {level }} \geq \max \left(\operatorname{dim}_{x}, \operatorname{dim}_{y}, \operatorname{dim}_{z}\right)$

## vtkOctreeScalars

This class is used to store the scalars in the model. The difference in this class, compared with vtkScalars is the indexing. The retrieval mechanism works by mapping the id through an associated vtkOctree as explained above in reference to figure 8.9.

The mapping of values in the class has to work exactly like the parent class for consistency. To achieve this, the fewer values stored in the vtkOctreeScalars structure is mapped to the entire volume through a vtkOctree. When a value is requested from the vtkOctree class, the call looks like

```
value = octree->GetPointData()->GetScalars()->GetScalar(ptId);
```

In the case of an octree, the results of the GetScalars() function call will return a vtkOctreeScalars object, that will know how to do the mapping. The id is one and the same, whether it is calculated for vtkStructuredPoints or vtkOctree structure.

The definition of vtkScalars has to be changed slightly, though to allow for the polymorphic form of the class to achieve the desired result. The change needed is to make the GetScalar() member function in vtkScalars a virtual function. Only after this change will vtkOctreeScalars operate as expected. The binding of GetScalar() is otherwise done at compile time and the redefined (overridden) GetScalar method would never be invoked at run time by classes that did not know about vtkOctreeScalars at compile time, which represents the whole toolkit as it is distributed.

This change was possible because of the open source nature of the toolkit.

## Class relationships

The classes in this implementation interact and rely on one another for their tasks. The relationships were indicated in figure 8.7. The figure describes how an octree object vtkOctree
has a vtkOctreeNode object, which is the head or root of the tree. The vtkOctreeNode has eight nodes of the same type, which are the subspaces, in case that the node is inhomogeneous. When the node is homogeneous, these children are not used and the object stores the scalarId instead.

The scalar values associated with a specific scalarId are stored in an associated vtkOctreeScalars class, which will hold values for the branches of the tree.

Because of the reduced number of scalars stored in the vtkOctreeScalars class, the use of the scalarId needs to be differentiated from the use of a pointId, unlike the case in the vtkStructuredPoints dataset. In the vtkOctree class, the indices indicated as $i d_{i}$ in figure 8.9 (to the left) correspond to the pointId values, while the ones labelled realId $d_{i}$ correspond to the scalarId values. For a developer using the class, though, the access mechanism combined with the overridden methods in vtkOctreeScalars make this transparent.

### 8.2.2 Creation of the octree structure

The algorithm implemented is a 3-D extension of Samet's algorithm to create quadtrees from binary arrays $[56]^{1}$.
procedure vtkOctree::Initialize(pts)
/* create the octree corresponding to a structured points object */
begin
int level;
this $\rightarrow$ SetDimensions ( pts $\rightarrow$ GetDimensions () );
$/^{*}$ calculation to have a cube of dimension $\left(2^{\text {level }}\right)^{3}$
large enough to hold the structuredPoints */
level $\leftarrow \operatorname{ceil}\left(\log _{2}\right.$ PowerOf2Dimension);
this $\rightarrow$ head $\leftarrow$ constructOctree ( level, $\left.2^{\text {level }}, 2^{\text {level }}, 2^{\text {level }}, \mathrm{pts}\right)$;
end

[^1]vtkOctreeNode procedure constructOctree(level, $\mathrm{i}, \mathrm{j}, \mathrm{k}, \mathrm{pts}$ )
$/^{*}$ construct the portion of an octree of size $\left(2^{\text {level }}\right)^{3}$
having $\mathrm{i}, \mathrm{j}, \mathrm{k}$ as the coordinates of the far corner of the subcube.*/
begin
nd $\leftarrow$ new vtkOctreeNode;
if (level $==0$ ) then
begin
/* process the voxel */
nd $\rightarrow$ HomogeneousOn();
/*find the scalar Id of the voxel and assign it in the node */
nd $\rightarrow$ SetScalarId(ArrayValue3D(pts, i-1, j-1, k-1));
end
else
begin
level $\leftarrow$ level $-1 ;$
half $\leftarrow 2^{\text {level }}$
nd $\rightarrow$ HomogeneousOff();
nd $\rightarrow$ node $[0] \leftarrow$ constructOctree(level, i - half, j - half, k-half, pts);
nd $\rightarrow$ node[1] $\leftarrow$ constructOctree(level, i, j - half, k-half , pts);
nd $\rightarrow$ node[2] $\leftarrow$ constructOctree(level, i - half, j, k-half , pts);
nd $\rightarrow$ node $[3] \leftarrow$ constructOctree(level, i, j, k-half , pts);
nd $\rightarrow$ node $[4] \leftarrow$ constructOctree(level, i - half, j-half, $\mathrm{k}, \mathrm{pts}$ );
nd $\rightarrow$ node[5] $\leftarrow$ constructOctree(level, $\mathrm{i}, \mathrm{j}$ - half, $\mathrm{k}, \mathrm{pts}$ );
nd $\rightarrow$ node $[6] \leftarrow$ constructOctree(level, i - half, j, k, pts);
nd $\rightarrow$ node $[7] \leftarrow$ constructOctree(level, $\mathrm{i}, \mathrm{j}, \mathrm{k}, \mathrm{pts})$;
/* check homogeneity */
$\mathrm{i} \leftarrow 1 ;$
boolean equal $\leftarrow \mathrm{nd} \rightarrow$ GetNode $(0) \rightarrow$ IsHomogeneous () ;
int previousId $\leftarrow$ nd $\rightarrow$ GetNode $(0) \rightarrow$ GetScalarId () ;

```
while (equal and i<8) do
begin
    equal }\leftarrow\textrm{nd}->\mathrm{ Getnode(i) }->\mathrm{ IsHomogeneous() and
        (previousId == nd }->\mathrm{ GetNode(i) }->\mathrm{ GetScalarId());
        i}\leftarrow\textrm{i}+1
        end
        if (equal) then /* all branches are of the same id*/
        begin
        nd}->\mathrm{ HomogeneousOn();
        nd }->\mathrm{ SetScalarId(previousId);
        for i := 0 to 7 do
        nd }->\mathrm{ Setnode(i,NULL); /* erase children*/
        end
    end
    return nd;
end
```


### 8.2.3 Test of memory usage and discussion

A test program was created to compare the memory usage of the octree implementation. The test consisted of reading a data file, creating the octree memory representation and reconstructing an structured points set from the octree model. Table 8.1 on page 145 shows the results. The only case in which the octree performs better than a standard structured points dataset is in the case of a regular pattern.

The performance of this octree implementation is poor. The overhead incurred in the management of a spatially stored data structure offset any savings in memory usage by storing only one value for an octree bucket.

The ratio between the memory used and the number of nodes of the octree remains constant at slightly over 64 (e.g. $72,839,000 / 1,138,089 \approx 64$ ). Clearly the number of bytes
used in a node is 64 and clearly most of the memory usage are pointers.
This storage mechanism is therefore generally not convenient because of the overhead in pointers.

The octree developed can reduce the amount of memory used to represent a volume data set. Jackins and Tanimoto[23] and Chen and Huang[9] present various algorithms for the creation and manipulation of these data structures. In previous work[43] it was stated that the octree scheme needed to be tested to decide whether it is worth applying the scheme. The tests here show that the overhead in pointers and the lack of large scale compression when all children have the same value leads to an inefficient storage structure.

| Data (filename) | Dimensions | File size | Structured <br> points | Octree | Number <br> of nodes |
| :---: | :---: | ---: | :--- | ---: | :--- |
| Corner elements (datz.vtk) | $8 \times 8 \times 8$ | 1,695 bytes | 1 kb | 10 kb | 137 |
| L Shape with bit values (tres.vtk) | $8 \times 8 \times 8$ | 1,414 bytes | 1 kb | 5 kb | 57 |
| L Shape - type long (tres-1.vtk) | $8 \times 8 \times 8$ | 1,416 bytes | 3 kb | 5 kb | 57 |
| Bits at random - type char (tres-2.vtk) | $100 \times 100 \times 100$ | $2,010,309$ bytes | $1,000 \mathrm{~kb}$ | $72,839 \mathrm{~kb}$ | $1,138,089$ |
| Regular pattern - type char (tres-3.vtk) | $100 \times 100 \times 100$ | $2,010,244$ bytes | $1,000 \mathrm{~kb}$ | 624 kb | 9,729 |
| Sierpinsky gasket - type char (tres-4.vtk) | $100 \times 100 \times 100$ | $4,227,316$ bytes | $2,096 \mathrm{~kb}$ | $70,277 \mathrm{~kb}$ | $1,098,057$ |
| Turbine blade (turb-17-01-2001.vtk) | $59 \times 120 \times 52$ | $1,103,995$ bytes | 369 kb | $2,469 \mathrm{~kb}$ | 38,561 |

Table 8.1: Memory usage of vtkStructured points and vtkOctree

## Chapter 9

## Voxel model

Based on the framework provided by the Visualization Toolkit (VTK), a system to support rapid manufacturing techniques is outlined by developing programs that permit the use of the model as indicated in figure 1.2.

### 9.1 Generating the model

### 9.1.1 VTK Datafile

In order to test some of the programs in this project, a voxel model was created directly 'by hand', because it is an expedite way of generating a voxel model.

The model is held as a structured points dataset within VTK.
The format used was the format of a standard VTK data file. An example file is displayed in figure 9.1. The values used inthe this example are integers of value 0,1 or 2 , which represent void space, material 1 and material 2 respectively.

The generation of the model from a data file is convenient for small sized models. The data file is created in a standard text editor. In the creation of the models, copy and past operations from a spreadsheet made some tasks easier. This method would not be suitable for larger models, where some other methods would be necessary, perhaps interactive or programatical input methods. The model on figure 9.2 was created by hand.
\# vtk DataFile Version 1.0
L shaped block with a through-shaft of material 2 - zero padded ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 101010
ORIGIN 0.0 0.0 0.0
ASPECT_RATIO 5.05 .05 .0

POINT_DATA 1000
SCALARS scalars float
LOOKUP_TABLE default
0000000000
$\begin{array}{llllllllll}0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0\end{array}$
0222222220
... one thousand data values $10 * 10 * 10=1000$
$\begin{array}{llllllllll}0 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 0\end{array}$
01001111111100
0000000000

Figure 9.1: Example VTK datafile


Figure 9.2: Voxel model generated manually

### 9.1.2 Semi automated model generation

A few tools and applications were linked to create a voxel model. The steps followed to create a representative voxel model are (see figure 9.3):

- Create a geometry in a solid modeller such as Unigraphics
- Transfer the geometry to a Finite Element package to do the meshing of the geometry.
- Transform the data from the mesh (tetrahedrons) to a data file for an unstructured grid in VTK.
- Assign values to the data points of the unstructured grid model to represent the appropriate material at the position.
- Sample the volume where the model is defined at a convenient sample rate and obtain a voxel model from the original model. The values asigned at the nodes of the unstructured grid are used in the interpolation.

This method was tested with a simple geometry (figure 9.4) and for the geometric data of a turbine blade. The geometry was obtained from Alstom (previously ABB Alstom Power) through the Rapid Manufacturing Consortium at De Montfort University.

The geometry was simplified, to include only the part corresponding to the foil of the turbine blade (figures 9.5 and 9.6 ), because of the simplification of geometry which makes it easier to process (transfer and mesh) the part through the finite element software. It is also expected that only the foil part of the turbine blade would present a gradation.

The simplified geometry, in the form of an IGES file was subsequently imported in Algor. A tetrahedral mesh is generated (see figure 9.7). Tetrahedrons are again selected because of their simple geometry that is more stable in the voxelization step.

The next step is to export the vertex and cell data. The feature to export directly the raw mesh information doesn't seem to be available in Algor, but the feature to export the data to ANSYS generates a text file with the information in ANSYS' prep7 format. The transformation of the prep7 data to vtk's format was done manually. A small change required


Figure 9.3: Alternative method of creation of a voxel model


Figure 9.4: A simple geometry used to test the transformations


Figure 9.5: Unigraphics presentation of the complete geometry of the turbine blade


Figure 9.6: Unigraphics presentation of the simplified geometry, i.e. foil section of the turbine blade


Figure 9.7: Algor presentation of the tetrahedral mesh generated


Figure 9.8: VTK pipeline to transform an unstructured grid (FEM mesh) to voxels
was the numbering of the nodes, which in prep7 format begins with one, had to be changed to vtk's convention to begin with zero.

The last step, which transforms the unstructured grid to voxels is done with a vtk pipeline implemented in a Tcl script. Figure 9.8 is a schematic of the pipeline.

The special component of this pipeline is the implicit data set, which takes the input from the unstructured grid. Its output is sampled by the next element in the pipeline. When queried for a value inside a tetrahedron in the unstructured grid, the implicit dataset interpolates the scalar values given at the nodes of the tetrahedral cell and returns the result to the sample function.

The result is then scaled with a vtkImageShiftScale filter to an integer to be written as an image by the last component of the pipeline, the vtkPNMWriter.

These image files have the same format as medical imaging data, and the result could have been visualized using Mimics, had the trial licence not expired.

### 9.1.3 Pipeline code

The code for the pipeline is outlined below:

```
vtkUnstructuredGridReader reader
reader SetFileName "../turbine100.vtk"
reader Update;
vtkImplicitDataSet dataset
dataset SetDataSet [reader GetOutput]
dataset SetOutValue 0
vtkSampleFunction theSample
    theSample SetImplicitFunction dataset
    theSample SetModelBounds -13.8 15.89 192.2 252.4 -13.43 12.65
    theSample SetSampleDimensions 59 120 52
    theSample ComputeNormalsOff
    theSample Update
    #the Update process queries the dataset 59*120*52 times
vtkImageShiftScale scale
    #calculate the shift and scale
    scale SetShift $shift
    scale SetScale $scale
    scale SetOutputScalarTypeToUnsignedChar
    scale SetInput [theSample GetOutput]
vtkPNMWriter pnmWriter
    pnmWriter SetFilePrefix "turbine"
```

```
pnmWriter SetInput [scale GetOutput]
pnmWriter Write
```


### 9.1.4 Discussion

The semi automated method used is much friendlier to the designer, because the geometry can be more conveniently represented using an advanced modeller, such as Unigraphics.

The step of conversion of the geometry to a tetrahedral mesh was not suitable for automatic execution. There were many problems in the conversion that had to be looked at durig the meshing. The acute edges of the foil seemed to have caused problems while meshing.

For the task of meshing, both ANSYS and Algor were used, selecting reasonable defaults and retrying several times to get a satisfactory solution. The meshing process with ANSYS could not be completed after several attempts varying many meshing parameters. Algor was successful after a few attempts, although it proved less flexible at the time of exporting the data of the resulting mesh.

This process was only tested using tetrahedral cells (figure 9.9).
The same transformation was performed on a simpler geometry (figure 9.4), to test the process with hexahedral cell elements (figure 9.10). These proved more problematic at the time of sampling through an implicit function and several 'cheese-holes' as shown on figure 9.11, appeared in the model sampled from hexahedra. The model sampled from tetrahedrons didn't have the same problem. The problem may be related to the treatment of hexahedral cells as they degenerate into tetrahedra by merging several nodes at a point.

### 9.2 Visualisation pipeline

In this section, an example pipeline is built to process a voxel model within the framework of VTK. The sequence of steps is represented in figure 9.12. This pipeline processes a model with two materials.


VTK_TETRA

Figure 9.9: Topology and numbering conventions for a tetrahedral cell in VTK


VTK_HEXAHEDRON

Figure 9.10: Topology and numbering conventions for a hexahedral cell in VTK


Figure 9.11: Mimics display of a voxel model of the solid shown in figure 9.4. The black regions show the problem associated with using hexahedra as opposed to tetrahedra.

### 9.2.1 Input step

The model data is stored in the data file ''dat.vtk'), which is read by the reader object to feed the pipe.

```
vtkStructuredPointsReader reader
    reader SetFileName "dat.vtk"
    reader Update
```


### 9.2.2 Classification step

The pipeline has two vtkImageThreshold objects: select1 and select2. The input to these objects is the output of reader, a ztypevtkStructuredPoints data consisting of zeros, ones and twos.
vtkImageThreshold select1
vtkImageThreshold select2
vtkMarchingCubes contour1
vtkMarchingCubes contour2

```
select1 SetInput [reader GetOutput]
    select2 SetInput [reader GetOutput]
```

The parameters assigned to the select objects filter the input to either material 1 or material 2 exclusively.

```
select1 ThresholdBetween 1 1
select1 SetInValue 1
select1 SetOutValue 0
select2 ThresholdBetween 2 2
select2 SetInValue 1
select2 SetOutValue 0
```

The output of the select objects is an structured points data set (vtkStructuredPoints) with only two values: either zero or one ("in" and "out" values), ie. the classified objects.

### 9.2.3 Contouring step

The next step is the contouring of the resulting volume. This is done with two contour objects. The class of these objects is vtkMarchingCubes.

```
vtkMarchingCubes contour1
vtkMarchingCubes contour2
    contour1 SetInput [ select1 GetOutput]
    contour1 SetValue 0 0.5
    contour2 SetInput [ select2 GetOutput]
    contour2 SetValue 0 0.5
```

The values chosen for the contouring are important. In this case the value 0.5 represents the mid-value between 0 and 1 , where the 1 represents material and 0 represents void. This values are linked to the choice of "in" and "out" values chosen for the classsification step.

### 9.2.4 Mapping

To complete the visualisation, the output of the contours is sent to two mappers, one for each contour. Actors are built in the standard way to display both materials. mapper 1 and mapper2 are both vtkDataSetMapper objects, which is a more general mapper used for visualisation. Since the data to be displayed in this case is of type vtkPolyData, mapper1 and mapper 2 could alternatively be of class vtkPolyDataMapper.

```
vtkDataSetMapper mapper1
vtkDataSetMapper mapper2
vtkActor actor1
vtkActor actor2
vtkRenderer ren
vtkRenderWindow renWin
```



Figure 9.12: Visualisation of the two contour surfaces obtained from the voxel model

```
mapper1 SetInput [contour1 GetOutput]
mapper1 ScalarVisibilityOff
mapper2 SetInput [contour2 GetOutput]
mapper2 ScalarVisibilityOff
```

actor1 SetMapper mapper 1
actor2 SetMapper mapper2
ren AddActor actor1
ren AddActor actor2


Figure 9.13: Visualisation of the model through slices
renWin AddRenderer ren

### 9.2.5 Visualisation through slices

A simple method to visualise a voxel model is through slices in the three coordinate planes. The pipeline for this case consists only of a reader of type vtkPNMReader and a viewer of type vtkImageViewer (figure 9.13). This pipeline presupposes the input in the form of image files.

### 9.3 Contouring for manufacturing

For the creation of the STL files that represent materials 1 and 2, a tcl/tk script implementing a custom-built pipeline was created. This is again a specific case of the visualization pipeline. The sequence of steps for processing a model with two materials is represented in figure 9.14.

The steps in this process are the same steps as in section 9.2: input, classification. contouring and mapping.

The first three steps, input, classification and contouring are exactly the same as the ones explained in section 9.2. In this case, though, one mapper of the class vtkSTLWriter is used for the mapping of the obtained contours.

```
vtkSTLWriter writer
```

    writer SetFileName "mat1.stl"
    writer SetInput [ contour1 GetOutput]
writer Write


Figure 9.14: Contouring the voxel model to create STL files. Two-material model.


Figure 9.15: Model manufactured from the voxel model

```
writer SetFileName "mat2.stl"
writer SetInput [contour2 GetOutput]
writer Write
```


### 9.4 Finite Element mesh generation

The finite element mesh for a given model is not unique, i.e. there are several possible meshes that represent the same model.

The following algorithm implements the generation of a simple mesh based on tetrahedral elements ${ }^{1}$. procedure CreateFiniteElementMesh(vm, nx, ny, nz)
/* create a mesh of tetrahedral elements based on the voxel model vm
with dimensions $n x, n y, n z^{*} /$
begin
/* generation of the nodes */

[^2]

Figure 9.16: Nodes for an element of the voxel model

```
int count }\leftarrow1\mathrm{ ;
for }\textrm{i}:=0\mathrm{ to nx
    for j:= 0 to ny
        for k:= 0 to nz
            GenerateNode(count, i, j, k, vm);
            count }\leftarrow\mathrm{ count + 1;
        end
    end
end
/* generation of the elements */
    int Xadd, Yadd, Zadd;
    int mat;
    int count }\leftarrow1\mathrm{ ;
/* node numbering for the first element (closest to origin) */
```

end
end
end

$$
\begin{aligned}
& \text { int } \mathrm{A} 0 \leftarrow 0 \\
& \text { int } \mathrm{B} 0 \leftarrow 1 \\
& \text { int } \mathrm{C} 0 \leftarrow \mathrm{nx}+1 \\
& \text { int } \mathrm{D} 0 \leftarrow \mathrm{nx}+2 \\
& \text { int } \mathrm{E} 0 \leftarrow(\mathrm{nx}+1)(\mathrm{ny}+1) \\
& \text { int } \mathrm{F} 0 \leftarrow(\mathrm{nx}+1)(\mathrm{ny}+1)+1 \\
& \text { int } \mathrm{G} 0 \leftarrow 2+2 \mathrm{nx}+\mathrm{ny}+\mathrm{nx} \mathrm{ny} \\
& \text { int } \mathrm{H} 0 \leftarrow 3+2 \mathrm{nx}+\mathrm{ny}+\mathrm{nx} \mathrm{ny} \\
& \text { for } \mathrm{i}:=0 \text { to } \mathrm{nx}-1 \\
& \text { Xadd } \leftarrow \mathrm{i}+1 ; \\
& \text { for } \mathrm{j}:=0 \text { to ny }-1 \\
& \text { Yadd } \leftarrow \mathrm{j}(\mathrm{nx}+1) ; \\
& \text { for } \mathrm{k}:=0 \text { to } \mathrm{nz}-1 \\
& \text { Zadd } \leftarrow \mathrm{k}(\mathrm{nx}+1)(\mathrm{ny}+1) \\
& \mathrm{A} \leftarrow \mathrm{~A} 0+\mathrm{Xdd}+\text { Yadd }+ \text { Zadd }+1 ; \\
& \mathrm{B} \leftarrow \mathrm{~B} 0+\mathrm{Xdd}+\mathrm{Yadd}+\text { Zadd }+1 ; \\
& \mathrm{C} \leftarrow \mathrm{C} 0+\mathrm{Xdd}+\text { Yadd }+ \text { Zadd }+1 \text {; } \\
& \mathrm{D} \leftarrow \mathrm{D} 0+\mathrm{Xdd}+\text { Yadd }+ \text { Zadd }+1 ; \\
& \mathrm{E} \leftarrow \mathrm{E} 0+\mathrm{Xdd}+\text { Yadd }+ \text { Zadd }+1 ; \\
& \mathrm{F} \leftarrow \mathrm{~F} 0+\mathrm{Xdd}+\mathrm{Yadd}+\text { Zadd }+1 ; \\
& \mathrm{G} \leftarrow \mathrm{G} 0+\mathrm{Xdd}+\text { Yadd + Zadd }+1 ; \\
& \mathrm{H} \leftarrow \mathrm{H} 0+\mathrm{Xdd}+\text { Yadd }+ \text { Zadd }+1 ; \\
& \text { ptId } \leftarrow \mathrm{vm} \rightarrow \text { ComputePointId( } \mathrm{i}, \mathrm{j}, \mathrm{k}) \text {; } \\
& \text { mat } \leftarrow \mathrm{vm} \rightarrow \text { GetPointData }() \rightarrow \text { GetScalars }() \rightarrow \text { GetScalar }(\text { ptId }) \text {; } \\
& \text { GenerateTetrahedralElements(count, A,B,C,D,E,F,G,H, mat); } \\
& \text { count } \leftarrow \text { count }+1 \text {; }
\end{aligned}
$$

end
procedure GenerateNode(count, i,j,k,vm)
/* Generate a node with node number count of structured coordinates i,j.k in the voxel model vm*/
begin
$\mathrm{x} \leftarrow \mathrm{vm} \rightarrow$ Origin $[0]+\mathrm{i}^{*}$ Spacing $[0] ;$
$\mathrm{y} \leftarrow \mathrm{vm} \rightarrow$ Origin $[1]+\mathrm{i}^{*}$ Spacing $[1]$;
$\mathrm{z} \leftarrow \mathrm{vm} \rightarrow$ Origin$[2]+\mathrm{i}^{*}$ Spacing $[2] ;$
write " N , count, $\mathrm{x}, \mathrm{y}, \mathrm{z}$ "; /* this is specific to ANSYS */
end
procedure GenerateTetrahedralElements(count, A,B,C,D,E,F,G,H, mat)
/* Generate six tetrahedral elements of material mat in the volume
defined by nodes A,B,C,D,E,F,G,H */
begin
write "mat $=$ ", mat; /*this is specific to ANSYS */
write "type $=72 " ; / *$ ANSYS tetrahedral element type is SOLID72 */
write "E", A,B,C,G
write "E", A,B,E,G
write "E", B,E,F,G
write "E", B,C,D,H
write "E", B, C,F,H
write "E", C,F,G,H
end


Figure 9.17: Programs for a system based on the voxel model

### 9.5 Summary

A system based on the voxel modelling technique can be developed based on the programs presented in this chapter, as described in figure 9.17.

The voxel model can be generated directly (manually), although this should prove tedious if not completely impractical. A more convenient method is presented in section 9.1 .2 , where the voxel model is generated by sampling an unstructured grid. This method can also be used to generate functional gradation in the volume by assigning values to the nodes of the unstructured grid model. This approach is similar to the cell-tuple method proposed by Jackson et al. (chapter 5). The sampling process in VTK assigns values interpolating within a cell using the scalar values assigned to the nodes of a cell.

The compression proposed using the octree was presented previously in chapter 8 .
Section 9.2 shows two methods of visualising the voxel model, either by contouring and mapping surface models or by visualising individual slices. The visualisation processes should prove easier to set up since we are building a system based on visualisation software.

Section 9.3 shows a method to produce industry standard STL files from a voxel model
through contouring. As an example, the STL files generated from a voxel model were produced using the SLA-7000 machine at the Rapid Manufacturing Group at De Montfort University (see figure 9.15).

## Chapter 10

## Summary, Conclusions and Future

## Work

This thesis studies the voxel modelling technique as means of representing multiple material objects and functionally graded material objects their possible realisation through Rapid Manufacturing techniques.

A review of Rapid Manufacturing processes was presented. The significant developments in the techniques are apparent as well as the industry acceptance of these methods for the reduction of design and manufacturing time cycles. It was shown that although the range of materials is limited, the techniques are suitable for the realisation of complex geometries with overhangs and undercuts that would be extremely difficult using conventional machining. Additive processes can also be used to create parts with varying composition.

The possible applications of Functionally Graded Materials were reviewed. Applications in aeronautics and astronautics show the importance of inhomogeneous materials in the future of engineering applications. Thermal Barrier Coatings (TBCs) consisting of graded layers of ceramic and metal have been used in many applications where parts are subject to high thermal stress, e.g. rocket engines, turbine blades. Other applications include cutting tools, optical fibres and biomaterials. The processing methods are often very elaborate and require special set-ups and sophisticated control of process parameters. RP methods capable of controlling the material distribution should simplify the creation of inhomogeneous objects.

Computational Geometry methods used in CAGD were also reviewed, covering first curve and surface modelling including the most general entities, the Non Uniform Rational BSplines (NURBS) for both curve and surface modelling as well as the IGES specification for the transfer of graphics and geometric data and Solid Modelling techniques. The voxel model is studied as a general case of volumetric data, considering various areas in science where this representations are useful. The memory requirements for a voxel model are considered in the light of the resolution available in FGM creation methods and Rapid Manufacturing processes. It is concluded that for some resolutions it is possible to consider the voxel modelling technique as a suitable method of representation.

The methods proposed by two research groups at the University of Michigan (Dutta et al.) and the M.I.T. (Jackson et al.) were reviewed. The first group proposes a method of representation of FGMs (inhomogeneous objects) using product manifolds and trivial fibre bundles. The researchers have an on-going work on the implementation of the 'heterogeneous solid modeller' based on the ACIS kernel, a commercial solid modelling kernel. The second group proposes the cell-tuple structure, based on the subdivision of the solid in cells. The first method is a more general and it encompasses the second approach (the cell-tuple subdivision).

The issue of the exploration of a usually large voxel model was considered. The creation of parts by RP methods from a voxel model, was likened to other visualisation task. The visualisation of a voxel model through volume rendering was studied and visualisation software for this type of application was identified. The Visualization Toolkit (VTK) was reviewed in this context and this toolkit was used in further work as centre of a voxel modelling system.

The issue of modelling an FGM by subdividing the object in finite elements within a FEA package such as ANSYS was considered and a trivial structural analysis example was modelled testing the system. The maximum resolution attainable using this commercial package was estimated at around 0.6 mm for this 2-D example.

The VTK software toolkit was revisited to extend it into a modelling system to support rapid manufacturing based on the voxel model. The design philosophy behind the toolkit and the concept of a visualisation pipeline were considered. Data representation within the framework of VTK was examined and the octree decomposition for graded materials was


Figure 10.1: Applications of the voxel model
implemented within the toolkit to consider its applicability as a compression mechanism to reduce memory requirements. The octree scheme was tested showing poor performance due to the overhead of pointer structures.

Finally the VTK framework was used to provide a system based on the voxel modelling method providing various transformations and possible uses. The programs developed give the outline of a system as indicated in figure 10.1. Two examples of application of the system were considered: a voxel model was generated from the geometry of a turbine blade and a part was created from two material model by contouring and built using the stereolithography process.

The developments in the Rapid Manufacturing processes have allowed the creation of FGMs by additive manufacturing. This fact added to the unique advantages of the processes e.g. the ability to create parts with geometry that would be extremely difficult to produce using standard subtractive methods present an interesting prospect for engineering applications.

The use of inhomogeneous objects in advanced engineering applications such as aerospace
and astronautics shows its importance for the future of engineering.
The voxel based techniques are particularly well suited for methods where the parts are created one layer at a time, (parallel/image based systems) such as Solid Ground Curing or Light Sculpting, where a whole slice is exposed simultaneously or material is deposited through a mask. These methods are not the norm currently, and there are far more sequential/vector based systems.

The voxel modelling technique was developed and tested based on the framework of a visualisation library. The problem of the limited resolution available was considered and tried to overcome using an octree decomposition of a graded material voxel model. This approach proved unsuccessful due to the large overhead of the structure. However the resolution may still be considered, because the rapid manufacturing processes based on powder processing by laser fusion or FGM creation by powder stacking are low resolution processes at the present stage.

There are still challenging tasks that remain to be solved. The system needs an extension to obtain a voxel model directly from a B-Rep representation or a surface model. The efforts in direct slicing of CAD models (Jamieson and Hacker[26]) are useful in this direction. They require however direct interaction with the solid modelling kernel, which in this case is Parasolid (the solid modelling kernel of Unigraphics). Another addition necessary for efficient creation of voxel models would be a CSG import utility, such as the on reported by Chandru[8] et al. as part of their Geometric Workbench for Rapid Prototyping (G-WoRP). These utilities could be added as an extension to the visualisation system framework, and it would require careful study of the proper abstraction of curve, surface and solid modelling constructs to achieve a good integration within VTK.

This integration of computational geometry methods in a visualisation framework may be in tune with the solution of problems cited by Farouki[17] and the desire for "new,open geometry engines" that would improve on the often unsatisfactory outcome of standards (e.g. IGES and STEP) when it comes to integrating CAD with other fields (e.g. CFD). Part of the problems in the transfer of solid models is that the development of CAD systems has been driven by product-release deadlines and while specific solutions may work sufficiently
well in industry, the application of CAD systems in challenging contexts inflicts pain and exasperation in the users. This difficulties are manifest in the lack of inter-operability among major CAD packages and the general difficulty in conveying all model information among systems and among applications.

The process planning of a powder deposition system suitable for processes such as ink jet printing or laser powder fusion needs to be tackled and integrated in a complete system. It was mentioned that at this stage models are created using the SDM process are "told" to use various materials. In another example of an inhomogeneous object built using the Sanders Model Maker machine (Kumar and Dutta[35]), the tool path for the material deposition was also generated manually. The paper states

This is a complicated problem and currently there does not exist an automated way of generating optimal tool paths for a given material distribution in a layer.

There is also the prospect of reexamining the use of voxel models at higher resolutions once the computing power becomes available, as there are other exacting applications which require massive computing power and have been forcing the limits of current computer architectures and configurations. A publicised example is the ambitious Grid Physics Network project[6] (http://www.griphyn.org), a system which will start to process petabytes of data per year.

## Appendix A

## Programs used for various tasks

## A. 1 Visualising a small voxel model

The following application of the VTK pipeline to display a voxel model was tested for a model of up to $10 \times 10 \times 10$ elements. The system creates polygonal data of a cube for every voxel present in the model using a glyph filter (vtkGlyph3D) and colours the voxel according to the its value.

## A.1.1 Tcl script

```
catch {load vtktcl}
source vtkInt.tcl
# program in tcl to display voxels as cubes and colour
# them according to their value
#
# Create the RenderWindow, Renderer and both Actors
vtkRenderer ren1
vtkRenderWindow renWin
    renWin AddRenderer ren1
vtkRenderWindowInteractor iren
    iren SetRenderWindow renWin
vtkStructuredPointsReader reader
    reader SetFileName "data.vtk"
    reader Update
vtkCubeSource cube
    cube SetXLength 0.5
    cube SetYLength 0.5
    cube SetZLength 0.5
```

```
vtkGlyph3D glyph
    glyph SetInput [reader GetOutput]
    glyph SetSource [cube GetOutput]
    glyph SetColorModeToColorByScale
    glyph SetScaleModeToDataScalingOff
vtkLookupTable lut
lut SetNumberOfColors 3
lut Build
#black and transparent
lut SetTableValue 0 0.0 0.0 0.0 0.0
lut SetTableValue 1 1.0 0.0 0.0 1.0
lut SetTableValue 2 0.0 0.0 1.0 1.0
vtkDataSetMapper mapper
    mapper SetInput [glyph GetOutput]
    mapper SetScalarRange 0 2
    mapper SetLookupTable lut
vtkActor ptsActor
    ptsActor SetMapper mapper
# Add the actors to the renderer, set the background and size
#
ren1 AddActor ptsActor
# render the image
#
iren SetUserMethod {wm deiconify .vtkInteract}
iren Initialize
```


## A.1.2 Example

The example is in the CD ROM in the directory 'glyphing'. By executing the tcl script glyph3.tcl, the system displays a glyph model of the data in the file in data.vtk. Once the program is executed, the user can change the view with the pointer in the render window. The user can also type 'u' to access the interactor dialogue.

The directory 'glyphing/comments' contains further comments and dialogue examples using the interactor.

## A. 2 Visualization of a model through slices

To visualise a larger voxel model, it becomes convenient to see the model by slices. The following scripts allow the visualisation of slices in any of the three orientations along the coordinate axes.

## A.2.1 Tcl script

The tcl script has three parts: rot-view.tcl, SliceOrder.tcl and WindowLevelInterface.tcl. These scripts are an adaptation of the "viewer" and the "frog" example that come with VTK.

```
rot-view.tcl
#
# Tcl script that displays the data from the series of PNM files
#
catch {load vtktcl.dll}
source "./SliceOrder.tcl"
vtkPNMReader reader
reader SetDataExtent 0 58 0 119 1 52
reader SetFilePrefix "./turbine"
reader SetTransform ap
reader ReleaseDataFlagOn
vtkImageViewer viewer
viewer SetInput [reader GetOutput]
viewer SetZSlice 14
viewer SetColorWindow 1100
viewer SetColorLevel }15
#viewer DebugOn
#viewer GetWholeZMin
#viewer GetWholeZMax
viewer Render
viewer SetPosition 50 50
#make interface
source WindowLevelInterface.tcl
```

```
WindowLevelInterface.tcl
# a simple user interface that manipulates window level.
# places in the tcl top window. Looks for object named viewer
#only use this interface when not doing regression tests
if {[info commands rtExMath] != "rtExMath"} {
# Take window level parameters from viewer
proc InitializeWindowLevelInterface {} {
    global viewer sliceNumber
    # Get parameters from viewer
    set w [viewer GetColorWindow]
    set l [viewer GetColorLevel]
    set sliceNumber [viewer GetZSlice]
    set zMin [viewer GetWholeZMin]
    set zMax [viewer GetWholeZMax]
    frame .slice
    label .slice.label -text "Slice"
    scale .slice.scale -from $zMin -to $zMax -orient horizontal \
        -command SetSlice -variable sliceNumber
    frame .wl
    frame .wl.f1
    label .wl.f1.windowLabel -text "Window"
    scale .wl.f1.window -from 1 -to [expr $w * 2] -orient horizontal \
        -command SetWindow -variable window
    frame .wl.f2
    label .wl.f2.levelLabel -text "Level"
    scale .wl.f2.level -from [expr $l - $w] -to [expr $l + $w] \
        -orient horizontal -command SetLevel
    checkbutton .wl.video
    # resolutions less than 1.0
    if {$w < 10} {
        set res [expr 0.05 * $w]
        .wl.f1.window configure -resolution $res -from $res -to [expr 2.0 * $w]
        .wl.f2.level configure -resolution $res \
-from [expr 0.0 + $l - $w] -to [expr 0.0 + $l + $w]
    }
    .wl.f1.window set $w
    .wl.f2.level set $l
    frame .ex
    button .ex.exit -text "Exit" -command "exit"
    pack .slice .wl .ex -side top
```

```
    pack .slice.label .slice.scale -side left
    pack .wl.f1 .wl.f2 -side top
    pack .wl.f1.windowLabel .wl.f1.window -side left
    pack .wl.f2.levelLabel .wl.f2.level -side left
    pack .ex.exit -side left
}
proc SetSlice { slice } {
    global sliceNumber viewer
    viewer SetZSlice $slice
    viewer Render
}
proc SetWindow window {
    global viewer video
    if {$video} {
        viewer SetColorWindow [expr -$window]
    } else {
        viewer SetColorWindow $window
    }
    viewer Render
}
proc SetLevel level {
    global viewer
    viewer SetColorLevel $level
    viewer Render
}
InitializeWindowLevelInterface
```

```
} else {
```

} else {
viewer Render
viewer Render
}

```
}
```


## SliceOrder.tcl

\#
\# these transformations permute medical image data to maintain proper orientation
\# regardless of the acqusition order. After applying these transforms with
\# vtkTransformFilter, a view up of $0,-1,0$ will result in the body part
\# facing the viewer.
\# NOTE: some transformations have a -1 scale factor for one of the components.
\# To ensure proper polygon orientation and normal direction, you must
\# apply the vtkPolyDataNormals filter.

```
#
# Naming:
# si - superior to inferior (top to bottom)
# is - inferior to superior (bottom to top)
# ap - anterior to posterior (front to back)
# pa - posterior to anterior (back to front)
# lr - left to right
# rl - right to left
#
vtkTransform si
[si GetMatrixPointer] SetElement 0 0 1
[si GetMatrixPointer] SetElement 0 1 0
[si GetMatrixPointer] SetElement 0 2 0
[si GetMatrixPointer] SetElement 0 3 0
[si GetMatrixPointer] SetElement 1 0 0
[si GetMatrixPointer] SetElement 1 1 0
[si GetMatrixPointer] SetElement 1 2 1
[si GetMatrixPointer] SetElement 1 30
[si GetMatrixPointer] SetElement 2 0 0
[si GetMatrixPointer] SetElement 2 1-1
[si GetMatrixPointer] SetElement 2 2 0
[si GetMatrixPointer] SetElement 2 30
[si GetMatrixPointer] SetElement 3 0 0
[si GetMatrixPointer] SetElement 3 1 0
[si GetMatrixPointer] SetElement 3 2 0
[si GetMatrixPointer] SetElement 3 3 1
vtkTransform is
[is GetMatrixPointer] SetElement 0 0 1
[is GetMatrixPointer] SetElement 0 1 0
[is GetMatrixPointer] SetElement 0 2 0
[is GetMatrixPointer] SetElement 0 3 0
[is GetMatrixPointer] SetElement 1 0 0
[is GetMatrixPointer] SetElement 1 1 0
[is GetMatrixPointer] SetElement 1 2 -1
[is GetMatrixPointer] SetElement 1 3 0
[is GetMatrixPointer] SetElement 2 0 0
[is GetMatrixPointer] SetElement 2 1-1
[is GetMatrixPointer] SetElement 2 2 0
[is GetMatrixPointer] SetElement 2 30
[is GetMatrixPointer] SetElement 3 0 0
[is GetMatrixPointer] SetElement 3 1 0
[is GetMatrixPointer] SetElement 3 2 0
[is GetMatrixPointer] SetElement 3 3 1
vtkTransform ap
ap Scale 1-1 1
vtkTransform pa
```

pa Scale 1-1 -1
vtkTransform lr
[lr GetMatrixPointer] SetElement 000
[lr GetMatrixPointer] SetElement 010
[lr GetMatrixPointer] SetElement $02-1$
[lr GetMatrixPointer] SetElement 030
[lr GetMatrixPointer] SetElement 100
[lr GetMatrixPointer] SetElement 1 1-1
[lr GetMatrixPointer] SetElement 120
[1r GetMatrixPointer] SetElement 130
[lr GetMatrixPointer] SetElement 201
[lr GetMatrixPointer] SetElement 210
[lr GetMatrixPointer] SetElement 220
[lr GetMatrixPointer] SetElement 230
[lr GetMatrixPointer] SetElement 300
[lr GetMatrixPointer] SetElement 310
[lr GetMatrixPointer] SetElement 320
[lr GetMatrixPointer] SetElement 331

## vtkTransform rl

[rl GetMatrixPointer] SetElement 000
[rl GetMatrixPointer] SetElement 010
[rl GetMatrixPointer] SetElement 021
[rl GetMatrixPointer] SetElement 030
[rl GetMatrixPointer] SetElement 100
[rl GetMatrixPointer] SetElement 1 1-1
[rl GetMatrixPointer] SetElement 120
[rl GetMatrixPointer] SetElement 130
[rl GetMatrixPointer] SetElement 201
[rl GetMatrixPointer] SetElement 210
[rl GetMatrixPointer] SetElement 220
[rl GetMatrixPointer] SetElement 230
[rl GetMatrixPointer] SetElement 300
[rl GetMatrixPointer] SetElement 310
[rl GetMatrixPointer] SetElement 320
[rl GetMatrixPointer] SetElement 331

## A.2.2 Example

The script displays slices of the voxel model in any of the three coordinate directions by changing the transformation that the vtkPNMReader uses, which can be one of ap. rl or is. These names stand for anterior-posterior (ap), right-left (rl) and inferior-superior (is). It is
important to tell the reader the size of the voxel model with the SetDataExtent command.
In this example the file prefix is set to turbine and the extent is set to $(0,58,0,119.1 .52)$. The last two values indicate the reader which files have to be read. This means that the reader expects to find the files turbine. 1 through turbine. 52 with images of 59 by 120 pixels. All image files have to be the same size.

## A. 3 Contouring a voxel model

## A.3.1 Tcl script

This script implements the conversion of a voxel model read from a vtk data file to a surface model (STL file). The pipeline for this program is described

```
#######################
#
# Program to contour a voxel model (isosurface extraction)
# Author: Ronaldo Mercado
#
#########################
# This program reads the data file 'datb.vtk'
# and produces an STL file for material 1 : 'mat1.stl'
# and another STL file for material 2: 'mat2.stl'
catch {load vtktcl}
source vtkInt.tcl
#
# pipeline : reader, select
#
vtkStructuredPointsReader reader
vtkImageThreshold select1
vtkImageThreshold select2
vtkMarchingCubes contour1
vtkMarchingCubes contour2
reader SetFileName "datb.vtk"
reader Update
select1 SetInput [reader GetOutput]
select2 SetInput [reader GetOutput]
select1 ThresholdBetween 1 1
select1 SetInValue 1
select1 SetOutValue 0
select2 ThresholdBetween 2 2
```

```
select2 SetInValue 1.5
select2 SetOutValue 0.5
contour1 SetInput [ select1 GetOutput]
contour1 SetValue 0 0.5
contour2 SetInput [ select2 GetOutput]
contour2 SetValue 0 1.0
#Display a pipeline
# uses mapper, actor, ren, renWin, iren
# add a colorbar actor
#
vtkDataSetMapper mapper1
vtkDataSetMapper mapper2
vtkActor actor1
vtkActor actor2
vtkRenderer ren
vtkRenderWindow renWin
vtkRenderWindowInteractor iren
vtkLookupTable lut
vtkScalarBarActor scalarBar
mapper1 SetInput [contour1 GetOutput]
mapper2 SetInput [contour2 GetOutput]
mapper1 ScalarVisibility0n
mapper2 ScalarVisibilityOn
lut SetHueRange 0 0.6667
lut SetSaturationRange 1 1
lut Build
mapper1 SetLookupTable lut
mapper2 SetLookupTable lut
scalarBar SetLookupTable lut
scalarBar SetTitle "Material"
actor1 SetMapper mapper1
actor2 SetMapper mapper2
ren AddActor actor1
ren AddActor actor2
ren AddActor scalarBar
renWin AddRenderer ren
iren SetRenderWindow renWin
iren Initialize
iren SetUserMethod {wm deiconify .vtkInteract}
```

wm withdraw .

```
#write the output of the pipeline to an STL file
# uses vtkSTLWriter
#
vtkSTLWriter writer
    writer SetFileName "mat1.stl"
writer SetInput [ contour1 GetOutput]
writer Write
writer SetFileName "mat2.stl"
writer SetInput [contour2 GetOutput]
writer Write
```


## A.3.2 Examples

The example is in the CD ROM in the directory 'contour'. Executing the tcl script contour.tcl generates the STL files of the voxel model stored in datb.vtk.

The second example processes the data in the file data.vtk which is the same voxel model without the padding on every side. This model doesn't generate closed surfaces and it is therefore not suitable for the creation of a solid.

## A. 4 Transformation from an unstructured grid into a voxel model

```
A.4.1 Tcl script
########################
#
# Program to transform from an unstructured grid into a voxel model
# Author: Ronaldo Mercado
#
##########################
# set up for the data set created for the simple geometry
#
# parameters:
#
#
#
set FILENAME "./ug-data.vtk"
set MODELBOUNDS "0 60 0 100 0 100"
```

```
set SAMPLEDIMENSIONS "60 60 60"
set MAXOUTPUTSCALAR 200
set OUTFILEPREFIX "outfile"
catch {load vtktcl}
vtkUnstructuredGridReader reader
eval reader SetFileName $FILENAME
reader Update;
vtkImplicitDataSet ids
ids SetDataSet [reader GetOutput]
ids SetOutValue 0
vtkSampleFunction theSample
    theSample SetImplicitFunction ids
    eval theSample SetModelBounds $MODELBOUNDS
    eval theSample SetSampleDimensions $SAMPLEDIMENSIONS
    theSample ComputeNormalsOff
    theSample Update;
vtkImageShiftScale uu
    set range [[[[theSample GetOutput] GetPointData ] GetScalars] GetRange]
    set bot [lindex $range 0]
    set top [lindex $range 1]
    set scale [expr $MAXOUTPUTSCALAR/($top-$bot)]
    uu SetShift [expr -$bot]
    uu SetScale $scale
    uu SetOutputScalarTypeToUnsignedChar
    uu SetInput [theSample GetOutput]
vtkPNMWriter pnmWriter
    eval pnmWriter SetFilePrefix $OUTFILEPREFIX
    pnmWriter SetInput [uu GetOutput]
    pnmWriter Write
```


## A.4.2 Example

The script is stored in the CD Rom in the directory 'voxel generation'. By executing the script task01.tcl, the system executes the steps:

1. Reads the an unstructured grid from the data file specified in the parameter FILENAME (e.g. set FILENAME "./ugdata.vtk").
2. Samples the volume specified in the parameter MODELBOUNDS at a sampling density given by SAMPLEDIMENSIONS.
3. Writes the resulting voxel model as a set of image files OUTFILEPREFIX.

It is important that the model bounds correspond to the data in the unstructured grid data file. The sampling is a computing intensive task that ties the processor for several minutes on the configuration used while testing ${ }^{1}$.

## A. 5 Octree implementation

The files for the octree implementation within VTK are located in the 'prgrams/octree' directory in the CD Rom. Table A. 1 lists the relevant files.

| Filename | Description |
| :--- | :--- |
|  |  |
| vtkScalars.h | Modified header file needed for the octree implementation |
| vtkOctree.h | Header file for the vtkOctree class |
| vtkOctree.cpp | Methods for the vtkOctree class |
| vtkOCtreeNode.h | Header file for the vtkOctreeNode class |
| vtkOctreeNode.cpp | Methods for the vtkOctreeNode class |
| vtkOctreeScalars.h | Header file for the vtkOctreeScalars class |
| vtkOctreeScalars.cpp | Methods for the vtkOctreeScalars class |
| main.cpp | Test procedures that visualise an octree |
| test.cpp | Test procedures that calculate the memory usage |
| *.vtk | Data files |

Table A.1: Files for the octree implementation in VTK

The nightly release of VTK is needed for the compilation of the procedures of test.cpp. because the release version does not have the GetActualMemorySize() methods. The libraries need to be compiled using the modified header file vtkScalars.h that specifies the GetScalar method to be virtual.

[^3]Both the release and nightly versions of VTK are included in the CD in the directories 'vtk' and 'vtknightly' respectively.

## Appendix B

## Object Oriented Programming

## Concepts

## B. 1 Principles of object oriented software

Object oriented systems have proved to be more modular, easier to maintain and to describe than traditional procedural systems. These advantages have been noticed by industry that has adopted object orientation for large projects. Several languages and metaphors have appeared over the years, from the Ada programming language in defence projects, Smalltalk developed by the Xerox PARC to the more recent Sun System's Java programming language. Alongside this evolution of programming languages, software development methodologies have also been evolving from the Object Modeling Technique (OMT) to the newer Unified Modeling Language (UML).

At the core of the huge field of software development methodologies lies the concept of representing a software system through computer abstractions that model physical or abstract pieces of the system being modelled. The dominating concept for this is the object that encapsulate both data and procedures i.e. properties and behaviour. The Object Oriented ( OO ) terminology and concepts are widely present in programming literature and textbooks that teach OO languages (e.g. [63]).

The terminology adopted in this document generally conforms to Rumbaugh's terminol-
ogy [28] and Stroustrup C++ specific terminology[59].
The characteristics of objects are:

- Identity. Each object has a unique handle within a computer program that makes it a discrete, distinguishable entity. Two objects are distinct even if all their attributes have the same value.
- Classification. Objects with the same data structure and behaviour are grouped into a class. This classification mechanism simplifies some tasks by allowing specialisation through subclassing, which creates a class hierarchy. Specific objects are instances of their class.
- Encapsulation. This characteristic refers to he data of an object being accesible only through well known methods or member functions, to enforce some standard interface of communication within objects. This characteristic is usually present in an object oriented languages.
- Polymorphism. Objects can exhibit a different behaviour for the same operation, depending on their class, which may implement the operation differently. For example. there may be a hierarchy of graphics objects which respond to the 'Draw' operation, a square and a circle would draw a different image although the operation requested (message passed) would be the same.
- Inheritance. Subclasses derived from other classes inherit attributes and behaviour from parent classes and implement specialised operations or add more attributes specific to the subclass. This serves as a method of factoring out common properties and operations.

The object oriented methodology as it is presented by Rumbaugh et al.[28] can be independent of the programming language, i.e. it should be possible to implement object oriented concepts on any high level programming language. such as C or Fortran. The sclection of a programming language will however have an impact on the implementation of the design
and usually the selection of an object oriented language will simplify and automate several tasks and controls for the developers by transferring responsibilities to the compiler.

## B. 2 Object Modeling Technique (OMT)

The OMT methodology proposes the development of a system model using three different viewpoints.

- The object model represents the static, structural, "data" aspects of the system.
- The dynamic model represents the temporal, behavioural, "control" aspects of a system.
- The functional model represents the transformational, "function" aspects of a system.

The basic concepts used for this project are classes, associations, aggregation and generalisation.

- Classes. Refers to an object class which describes a group of objects with similar properties (attributes), common behaviour (operations), common relationships to other objects and common semantics.
- Associations. Refer to relationships that exist among objects and classes.

Associations have a certain multiplicity that specifies the number of objects in a class at either end of the association.

Associations are usually represented by pointers that may get confused as attributes. An association does not exist in a class if not in relation with some other class.

- Aggregation. This is a special type of association from a class that represents the whole, the assembly class, to the classes that represent the parts.
- Generalization (inheritance). This abstraction mechanism allows to factor out common characteristics of classes and preserve the differences. The more general class is the superclass, and the specialised or refined version is called a subclass.

Class:

| Class Name |
| :--- |
| attribute: <br> attribute: data_type <br> attribute: data_type $=$ init_value |
| operation() <br> operation(arg_list:) <br> operation(arg_list:) : return_type |



Figure B.1: Basic Object Model Notation

The OMT methodology includes a convenient graphical representation for these concepts (figure B.1). This description language is the base of the more recent UML methodology[20]. which is found in CASE tools and drawing programs because of its popularity.

Classes are represented with boxes that may include the attributes and operations. Associations are represented with lines. The multiplicity of an association is represented with numerals or circles at either end of the relationship. Inheritance is represented with a triangle on the side of the more general class. Aggregation presents a diamond at the end of the assembly class.

## Appendix C

## Results of the corner bracket example

The results for the corner bracket example described in chapter 7 are included in this appendix. Figure C. 1 shows the stress distribution for the bracket modelled assuming only one material (steel). The materials are assigned for the following step according to the average stresses in the elements. This is shown in figure C.2. Finally, figure C. 3 shows the stress distribution for the same bracket geometry for the five material model. Table C. 1 shows the material properties used in the example.

| Material | Young's modulus [Pa] | Poisson's ratio |
| :---: | :---: | :---: |
|  |  |  |
| 1 | $205.00 \times 10^{9}$ | 0.27 |
| 2 | $171.25 \times 10^{9}$ | 0.29 |
| 3 | $137.50 \times 10^{9}$ | 0.31 |
| 4 | $103.75 \times 10^{9}$ | 0.33 |
| 5 | $70.00 \times 10^{9}$ | 0.35 |

Table C.1: Material properties used in the example bracket. Material 1 has the properties of steel and material 5 has the properties of aluminium



Figure C．2：Material assignments


## Appendix D

## Publication

Voxel Modelling for<br>Rapid Manufacturing<br>R. Mercado, J.M. Blackledge and P. Dickens

Fourth International Scientific Colloquium CAx Techniques, Bielefeld Germany. September 1999.

# Voxel Modelling for Rapid Manufacturing 

Mercado R. ${ }^{\text {a }}$, Blackledge J.M. ${ }^{\text {a }}$, Dickens, P. ${ }^{\text {b }}$<br>${ }^{\mathrm{a}}$ Faculty of Computing Sciences and Engineering, Department of Mathematical Sciences. Institute of Simulation Sciences, De Montfort University, Leicester, LE1 9BH. UK<br>${ }^{\mathrm{b}}$ Faculty of Computing Sciences and Engineering, Rapid Manufacturing Research Group. De Montfort University, Leicester, LE1 9BH, UK


#### Abstract

Voxel modelling refers to dividing up three-dimensional space into cubical cells at a particular resolution. Objects are modelled by listing the cells that they occupy. This scheme of representation requires large amounts of core memory for reasonable resolution and thus has not been generally favoured for practical systems. This article is a literature survey on voxel modelling in general and an investigation of methods used to represent objects with a variable material composition. Potentially these could be fabricated using Rapid Manufacturing, a family of technologies that generate three-dimensional, solid objects under computer control. The paper explores relevant areas to the subject such as rapid manufacturing, solid modelling and computer graphics, volume rendering and scientific visualisation. Examples of materially graded objects created using Rapid Manufacturing technologies are reviewed. Graphical display techniques available for voxel modelling are reviewed and two public domain utilities for volume rendering are tested. Some of the 3-D processing techniques that could be used to compress voxel based models are considered. It is concluded that current modelling of these objects lags behind the realisation of objects in practice. The article reports current research and explores possible future research areas.


Keywords: Rapid Manufacturing, Functionally Gradient Material applications, Voxel Modelling.

## 1. Introduction

Rapid Manufacturing is a family of technologies that generate three-dimensional, solid objects under computer control with three important features in common:

- Parts are automatically produced from CAE data sets under computer control.
- These techniques are "additive": an object is built by successively adding raw material. rather than removing existing material, which is the case with production techniques such as milling.
- A set of layers or "slices" are added together to create a solid volume of the desired shape.

Parts produced by major commercial Rapid Manufacturing systems are made of a single material, although some of these techniques are potentially capable of handling multiple materials. Three papers already report on objects with material gradation in the volume $[15,21,30]$.

Current rapid manufacturing applications traditionally use industry standard solid modellers. The model is created using standard CSG and B-Rep modellers. The internal volume is assumed to be filled with a homogeneous, isotropic material. This model is then tessellated and transferred to the rapid manufacturing apparatus of choice. This scheme has been in use for close to a decade, and although the size of STL files for complex object may be enormous in size (e.g. 100 MB ), these are still manageable using current computer technology.

None of the traditional solid modelling strategies can represent materially graded objects, they only capture their geometry and topology [14, 16, 17].

## 2. Voxel modelling

### 2.1. Volume data sets representation and application

A number of techniques have been developed to represent volumetric data. Volume data sets are typically sets $S$ of samples ( $\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{v}$ ), representing the value v of some property of the data at a 3-D location ( $x, y, z$ ). In general, the samples may be taken at random locations in space, but in many cases $S$ is isotropic, containing samples taken at regularly spaced intervals along three orthogonal axes. Since S is defined on a regular grid, a 3-D array is typically used to store the values. A function may be defined to describe the value at any continuous location by approximating v at a location ( $\mathrm{x}, \mathrm{y}, \mathrm{z}$ ) using some interpolation function to S . The region of constant value that surrounds each sample in zero-order interpolation is known as a volume cell (voxel for short), with each voxel being a rectangular cuboid.

In addition to regular grids, rectilinear, curvilinear, and unstructured grids are employed. In an unstructured grid, there is no explicit or implicit grid topology. Unstructured grids are common for scattered data, finite-element/volume analysis, and computational fluid dynamics.

The primary sources of volume data sets are three: sampled data of real objects or phenomena, computed data produced by a computer simulation, and modelled data from a geometric model. Volume visualisation allows the user to extract information from volumetric data sets through interactive graphics and imaging. The importance of the voxel model in medical imaging [19] comes from its use in the CT, MRI, SPECT, and PET medical imaging modalities as well as for rendering 3-D medical images.

### 2.2. Memory Requirements

A voxel model requires huge amounts of memory. To achieve a good resolution (e.g. $5 \mu \mathrm{~m}$ ) in a considerable volume (e.g. $500 \mathrm{~mm} \times 500 \mathrm{~mm} \times 500 \mathrm{~mm}$ ) requires $100,000^{-}$ voxels, i.e. $10^{15}$ elements. To grasp the enormity of this figure, according to some rough estimates, the information of all U.S. academic libraries together is twice that amount, roughly $2 \times 10^{15}$ bytes [21]. The largest volume data set that current high-end systems can handle is a $1024 \times 1024 \times 1024$ (roughly $10^{9}$, a gigabyte) element data set using hardware optimised for 3-D graphics.

The storage requirement of a voxel model is $n^{3} \sum_{i=1}^{n_{p}} p_{l}$ where $n_{p}$ is the number of properties and $p_{l}$ is the storage requirement of a value of the property $l$. Typical voxel models in medicine are based on a value of $\mathrm{n}=512$, and store a single density property represented by an integer. In this case, the voxel model occupancy is around 512 MB . In other application areas, such as in earth sciences, the memory storage could be increased by 10 to 50 times. This is the major drawback of voxel models [1].

### 2.3. Compression Methods

In principle, the compression methods in the 3-D domain are a generalisation of the compression methods available in 2-D for working with raster images. Some compression methods are compression based on the DCT (Discrete Fourier Transform), compression based on wavelets, fractal compression, multiresolution representations and compression based on hierarchical structures (Octree and BSP trees) [17].

Many of these techniques have been studied in relationship with their application for medical imaging and volume rendering. In fact medical imaging equipment often uses either the raw voxel model or a octree model for the visualisation [19].

The octree representation uses a recursive subdivision of the space of interest into eight octants that are arranged into an 8 -ary tree (hence the name). This type of structure is analogous to the quadtree, which is used in 2-D raster image processing. The octant volumes continue to be subdivided until a termination criterion is satisfied. Two common termination criteria are the total volume represented by a node and the complexity (homogeneity) of the volume represented by the node.

In general, the number of nodes in this type of octree representation of a solid object is proportional to the surface area of the object. Hence octree models are not quite as large as exhaustive representations but still take a fair amount of storage [16]. An isosurface octree, or the classical octree of a voxel mode defines voxels as black if their associated value is within a specific range of the property and white otherwise. The voxels whose property values are within this range and differ less than a given $\varepsilon$ are recursively grouped into black nodes. This type of octree is only useful when the volume is not very heterogeneous [1].

### 2.4. G-WoRP, a hybrid voxel modeller

Chandru et al.[2] describe G-WoRP, a Geometric Workbench for Rapid Prototyping. In this novel computer-aided design tool, the authors extend the traditional solid modelling hybrid model architecture to include the voxel and the slice, the real manufacturing primitives for Rapid Prototyping (Rapid Manufacturing) systems. The paper concludes with the status of this tool by the time of publication (1994), when the implementation of a prototype was in progress for the Silicon Graphics platform. Although the work on G-WoRP as well as other related tools has been reported in several Masters Project reports, it has not yet been published [17].

## 3. Materially graded objects

Materially graded objects are objects composed of different constituent materials and could exhibit continuously varying composition and/or microstructure. Such continuous changes result in gradation in their properties. Materially graded objects are potentially ideal for several engineering applications.

### 3.1. Applications

By creating objects with spatially varying material properties, one can tailor the composition of an artefact such that material properties match the functional requirements demanded of the component at a given point. For example, for optimal tool life it is desirable to have a hard outside shell for wear resistance and a ductile core to resist brittle fracture. Traditionally, such benefits have been achieved through the coating or cladding of existing artefacts with shells of different physical characteristics. Surface treatment methods are similar in purpose. These methods change basically only the surface hardness. Material gradation may, however, change other properties besides the hardness, e.g. the thermal conductivity or the density. Another possible application could be the construction of an engine block, which could have arbitrarily-shaped cooling channels. The walls of the cooling channels could mostly comprise a high thermal conductivity material to aid heat transfer [17].
"Project Maxwell" [5] discusses several ideas for the possible applications of multiple material objects and functionally graded materials. It discusses the concurrent design of the structure and the material by creating micro-scale voids where a structure is not required to support loads. The next step is the creation of composites, inserting materials that can improve strength, toughness, vibrational characteristics, acoustics, impact resistance and energy absorption.

Non-homogeneous composite materials result in significant improvements in thermo-mechanical properties without increase in weight. A design criterion such as bending rigidity can be dramatically improved using composites with a stronger material in the outer surfaces and weak and lighter materials in the inner core. The authors also suggest applications in automobile panel design, introducing complex micro-structures, whose plastic deformation can absorb large amounts of energy. This feature may be very advantageous in side panels for side impact protection.

### 3.2. Example objects

Fessler et al.[6], Jepson et al.[7] and Kumar[14] have produced objects of varying composition throughout the volume using Rapid Manufacturing technology.

The Shape Deposition Manufacturing (SDM) process permits the creation of multimaterial structures and optional embedded electronic components [20]. Fessler et al. used an improved SDM system that enables the deposition of functionally graded metals through the use of powder mixing. The addition of powder mixing enables the deposition of single layers in which material properties can be smoothly varied without discrete interfaces between dissimilar materials. It has been shown that certain materials will completely mix during deposition and form alloys that exhibit properties intermediate to those of the constituent feed powders.

An example of a materially graded object was created with this method. The object is an advanced ALCOA moulding tool. The tool is made of Invar, stainless steel and copper and has two cooling channels in each half to remove heat quickly from the part.

Jepson et. al. used an addition to the Selective Laser Sintering (SLS) process, a process known as $\mathrm{M}^{2}$ SLS, which enables the fabrication of materials with varying material composition. The process has been tested with tungsten carbide and cobalt, a ceramic/metal combination, with potential applications as a cutting tool.

Kumar and Dutta built an object using the Sanders Model-maker. A probe of smoothly varying volumetric fraction was built by modifying the tool path generation strategy. Given a certain layer distribution, there is currently no method for the automated generation of an optimal tool path for its fabrication.

### 3.3. Modeling Materially Graded Objects

To model a functionally graded material (FGM) object using exhaustive enumeration is just a matter of representing in every cell of the volume a value v which represents the material composition. Modelling the volumetric fraction of a composite object at a fine resolution is the goal of the modelling technique, which would allow heterogeneous objects to be manufactured using layered techniques.

To determine a proper range and resolution of the volumetric fraction for rapid manufacturing applications, two test objects reported by Jepson et al. [7] and Fessler et al. [6] were examined. In the SDM example, a smooth variation of properties can be achieved with a relatively coarse variation of material composition. Varying the composition in $1 \%$ steps is a satisfactory resolution for practical foreseeable engineering applications.

## 4. Graphical Display Techniques

To be able to interact with a multi-material model, the designer would have to obtain a visual representation of the model on a display. The volume rendering approach seems the best suited for the task. Volume rendering is a method used to capture an entire 3-D data set in a 2-D image directly from the volumetric data. Volume rendering differs from traditional computer graphics, which simulate a scene by rendering surfaces of a model of basic building blocks such as cylinders, spheres, planes, points and polygons.

Volume rendering also differs from image processing in that although the process may need image enhancing, filtering and other typical image processing techniques there is a 3-D data set to work with. The emphasis of volume rendering is the interior, which cannot be captured in an image by simple surface rendering.

There are several "off the shelf" volume rendering applications available. There are many for medical imaging applications while some other are application specific.

Two public domain systems were tested [17]: VolVis and GVLware (BoB). VolVis, developed at the State University of New York at Stony Brook, is a comprehensive volume visualisation system available in several platforms and as source code. The GVLware is a public domain application developed by the Army High Performance Computing Research Center (AHPCRC).

The volume visualisation systems are greedy on resources. The VolVis system works on several platforms and it was possible to compile it on a HP workstation running HP-UX 9. The system was tested on two types of displays: an 8 -bit colour display and a 24 bit colour display. The resolution is poor using only 256 colours ( 8 -bits) and so 24 -bit colour is mandatory for colour volume rendering.

## 5. Alternative Representation Methods

Kumar and Dutta have initially presented a method to represent multi-material objects [11] and later have extended this method to represent functionally graded materials [12, 14]. They propose a new mathematical model for the representation of multiple materials.Their discussion uses concepts of point-set topology, which is a convenient mathematical method to characterise rigorously the properties of three-dimensional objects. To be able to represent multiple materials, a material dimension $\mathbf{M}$ is included, apart from the spatial dimensions $\mathbf{R}^{\mathbf{3}}$ that capture the geometry and topology of an object. For a finite number of unique materials, the choice for the material dimension $\mathbf{M}$ would be the set of integers $\mathbf{I}$. Then the product space $\mathbf{T}=\mathbf{R}^{3} \times \mathbf{I}$ with the product topology can form a new modelling space for representing multiple-material objects.

A solid described using traditional solid modelling techniques is a member of the class of r-sets $\mathbf{A}$ in $\mathbf{R}^{3}$. The method proposes a new class $\mathbf{A}_{\mathbf{m}}=\mathbf{A} \times \mathbf{K}$, where $\mathbf{A}$ is the class of $r$-sets and $\mathbf{K} \subset \mathbf{I}$ is a finite set of integers. Each material is characterised by an integer in $\mathbf{K}$. A typical member $Q \in \mathbf{A}_{\mathbf{m}}(Q=\{P, k\})$ is called a $r_{m}$-set and is composed of an $r$-set $\mathrm{P} \in \mathbf{A}$ and an integer $k \in \mathbf{K}$.

This definition is extended to represent functionally graded materials. To model objects with continuous material variation, the material space must be expanded from $\mathbf{K} \subset \mathbf{I}$ in the previous case. A suitable choice for the new mathematical space is $\mathbf{T}=\mathbf{R}^{3} \times \mathbf{R}^{\mathbf{n}}$, n being the number of primary materials. $\mathbf{R}^{3}$ is the geometry space, where geometry and topology are defined, using a traditional solid model. $\mathbf{R}^{\mathbf{n}}$ is the material space. The material can be identified at any point by volume fractions of each of the primary materials. Each point in an object $S$ can now be characterised in product space
$\mathbf{T}$ as $(\mathbf{x}, \mathbf{v}(\mathbf{x}))$ where $\mathbf{x} \in S$ is a point in the object and $\mathbf{v}(\mathbf{x}) \in \mathbf{R}^{\mathbf{n}}$ represents the material at that point. This work is extensive and mathematically rigorous. The authors recognise however some blanks, which are still left to research.

Some other ideas were developed for the SDM process mentioned in section 3.2. The usual subdivision in layers is not enough in this case, since each layer may have more than one material. The concept of compacts is introduced, as a further subdivision of a 3-D layer. Compacts can have partitions along surfaces whose normals are not necessarily along the build-up direction. The SDM process has always had the ability to create multi-material objects.

The additions to this process, reported by Fessler et al., allow metallic powders from different powder feeders to mix under a laser, however the function gradient aspect of the tool created with these additions, was not modelled at all. Basically, the model thought it was one material and the deposition files were modified by hand to make the material transition [17].

## 6. Conclusions

The survey of methods to represent objects with a variable material composition has explored a number of relevant areas: rapid manufacturing, solid modelling, computer graphics, volume rendering and scientific visualisation. Some conclusions can be drawn from this survey:

There is currently no established method to represent materially graded objects, although there are efforts in this direction, notably the method proposed by Kumar and Dutta.

The realisation of these objects through Rapid Manufacturing is ahead of the representation methods delivered by computational modellers.

Voxel representation techniques have all the potential to deliver Rapid Manufacturing representation requirements. However the huge memory requirements make it stumble as a high-resolution representation method.

From a detailed look at the modelling requirements for functionally graded objects, using the example objects, it is apparent that a material volumetric fraction variation of $1 \%$ is able to accommodate foreseeable engineering applications.

There are several volume-rendering applications available both commercially and in the public domain. These applications are capable of working with data sets up to around $1024 \times 1024 \times 1024$ elements.

## 7. References

1. Brunet, P. et al.: Modeling and visualization through data compression. In: Rosenblum, L. et al., (eds.): Scientific Visualization: Advances and challenges, Academic Press, London, 1994. pp. 157-169
2. Chandru, V., Manohar, S.: G-WoRP: a geometric workbench for rapid prototyping. Manufacturing Science and Engineering.vol. 2 (1994), pp.569-574.
3. Chandru, V. et al.: Voxel-based modeling for layered manufacturing. IEEE Computer Graphics and Applications, vol. 15 (1995), no. 6, pp.42-47
4. Chandru, V., Manohar, S.: Volume modeling for emerging manufacturing technologies. Sadhana, Journal of the Indian Academy of Sciences, Fall 1997.
5. Dutta, D. et al.: Project MAXWELL: Towards rapid realization of superior products. In: Proceedings of the 1992 Solid Freeform Fabrication Symposium University of Texas at Austin, 1992. pp.54-61
6. Fessler, J. et al.: Functional gradient metallic prototypes through Shape Deposition Manufacturing. In: Proceedings of the 1997 Solid Freeform Fabrication Symposium University of Texas at Austin, 1997. pp.521-528
7. Jepson, L. et al.: SLS processing of functionally gradient materials. In: Proceedings of the 1997 Solid Freeform Fabrication Symposium. University of Texas at Austin, 1997. pp.67-79
8. Kaufman, A. et al.: Volume Graphics. Computer, vol. 26 (1993) no.7, pp.51-64.
9. Kaufman, A.: Volume visualization. ACM Computing Surveys, vol. 28 (1996) no.1, pp. 165-167
10. Kumar, V. et al.: Rapid Design and Prototyping of Customized Rehabilitation Aids. Communications of the ACM, vol. 39 (1996), no.2.
11. Kumar, V., Dutta, D.: An approach to modeling multi-material objects. In: Proceedings of the 4th ACM Symposium on Solid Modeling, ACM, Atlanta GA, May 1997.
12. Kumar, V., Dutta, D.: Solid model creation for materially graded objects. In: Proceedings of the 1997 Solid Freeform Fabrication Symposium University of Texas at Austin, 1997. pp.613-620
13. Kumar, V., Dutta, D.: An assessment of data formats for layered manufacturing. In: Advances in engineering software, vol. 28 (1997), pp.151-164
14. Kumar, V., Dutta, D.: An approach to modeling \& representation of heterogeneous objects. Paper in review, ASME Journal of Mech. Design (May 1998).
15. Lichtenbelt, B. et al.: Introduction to volume rendering Prentice Hall, 1998.
16. Mäntylä, M:.An introduction to solid modeling Rockville MD, Computer Science Press, 1988.
17. Mercado, R.: Voxel Modelling for Rapid Manufacturing. MSc in CAE Techniques Project Report, Faculty of Computer Sciences and Engineering, De Montfort University, Leicester UK, 1998.
18. Sanders Prototype Inc.: Sanders Prototype Inc. Web site [online], 1997. Available at http://www.sanders-prototype.com/main.html [Accessed August 1998]
19. Stytz, M. et al.: Three-Dimensional Medical Imaging: Algorithms and Computer Systems ACM Computing Surveys, vol. 23 (1991), no. 4 pp. 421-499.
20. Weiss, L., et al.: Shape Deposition Manufacturing of Heterogeneous Structures. Journal of Manufacturing Systems, vol. 16. (1997), no.4, pp.201-227.
21. Williams, R.: Data Powers of Ten [online], No date. Available from: http://www.ccsf.caltech.edu/~roy/dataquan/ [Accessed 04 Aug 1998].

## References

[1] R. Avila et al. Towards a comprehensive volume visualization system. In A. Kaufman and G. Nielson, editors, Visualization '92, pages 13-20, Boston, MA, October 1992. IEEE Computer Society Press.
[2] R. Avila et al. Volvis: A diversified volume visualization system. In IEEE Visualization Proceedings, pages 31-38, Washington DC, October 1994. IEEE Computer Society Press.
[3] S. Bhashyam, K.Hoon, and D.Dutta. An integrated CAD system for design of heterogeneous objects. Rapid Prototyping Journal, 6(2):119-135, 2000.
[4] J.M. Blackledge. Product data exchange, lecture notes. Department of Mathematical Sciences, De Montfort University, 1996.
[5] P. Brunet et al. Modeling and visualization through data compression. In L. Rosenblum et al., editors, Scientific visualization: Advances and challenges, pages 157169. Academic Press, 1994.
[6] S. Cass. Super nets for supercomputers. IEEE Spectrum, 38(1), January 2001.
[7] Castle Island Co. Commercial rapid prototyping system manufacturers. [online] http://home.att.net/~castleisland/com_lks.htm, August 2000. Accessed Aug 2000.
[8] V. Chandru et al. Voxel-based modeling for layered manufacturing. IEEE Computer Graphics and Applications, 15(6):42-47, 1995.
[9] H. Chen and T. Huang. A survey of construction and manipulation of octrees. Computer Vision and Image Processing, 43:409-431, 1988.
[10] P. Cignoni et al. Multiresolution modeling and visualization of volume data based on simplicial complexes. In Proceedings of the ACM symposium on volume visualization. pages 19-26, Washington, 1994. ACM Press.
[11] P. Cignoni et al. Multiresolution representation and visualization of volume data. IEEE Transactions on visualization and computer graphics, 3(4):352-369, 1997.
[12] W. Cochran et al. Fractal volume compression. IEEE Transactions on visualization and computer graphics, 2(4):313-321, 1996.
[13] Microsoft Corp. MSDN library. [CD-ROM], 1998. Visual Studio 6.0 Edition.
[14] Cubital Inc. Cubital. [online] http://www.cubital.com/index.html, 1997. Accessed August 2000.
[15] A. Dolenc. An overview of rapid prototyping technologies in manufacturing. [online] http://www.cs.hut.fi/~ado/rp/rp.html, July 1994. Accessed Aug 2000.
[16] G. Farin. Curves and surfaces for computer aided geometric design: a practical guide. Academic Press, 2nd edition, 1990.
[17] R. Farouki. Closing the gap between CAD model and downstream application. SIAM News, 38(5), 1999.
[18] J. Fessler et al. Functional gradient metallic prototypes through shape deposition manufacturing. In Proceedings of the Solid Freeform Fabrication Symposium. University of Texas at Austin, August 1997.
[19] S. Finger et al. Rapid design and manufacture of wearable computers. Communications of the $A C M, 39(2)$, February 1996.
[20] M. Fowler. UML distilled: applying the standard object modeling language. AddisonWesley, 1997.
[21] Y. Horstein. 3D systems cross-license agreement in japan. [online] http://ltk.hut.fi/archives/rp-ml-1999/0066.html, Jan 1999. Accessed Aug 2000.
[22] IBM. Open visualization data explorer. [online] http://www.research.ibm.com/dx/, No date. Accessed October 2000.
[23] C. Jackins and S. Tanimoto. Oct-trees and their use in representing three-dimensional objects. Computer Graphics and Image Processing, 14:249-270, 1980.
[24] T. R. Jackson, H. Liu, N. M. Patrikalakis, E. M. Sachs, and M. J. Cima. Modeling and designing functionally graded material components for fabricatino with local composition control. Materials and Design, 20:63-75, 1999.
[25] P. F. Jacobs. Rapid Prototyping and Manufacturing. Fundamentals of Stereolithography. Mc Graw Hill, 1992.
[26] R. Jamieson and H. Hacker. Direct slicing of cad models for rapid prototyping. Rapid Prototyping Journal, 1(2):4-12, 1995.
[27] L. Jepson et al. SLS procesing of functionally gradient materials. In Proceedings of the Solid Freeform Fabrication Symposium. University of Texas at Austin. August 1997.
[28] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, and W.Lorensen. Object-Oriented Modeling and Design. Prentice-Hall, 1991.
[29] A. Kaufman. Volume visualization. ACM Computing Surveys, 28(1):165-167, 1996.
[30] Inc. Kitware. http://www.kitware.com. Web site. Accessed August 2000.
[31] O. König and G. Fadel. Application of genetic algorithms in the design of multi-material structures manufactured in rapid prototyping. In Proceedings of the Solid Freeform Fabrication Symposium, pages 209-217. University of Texas at Austin, August 1999.
[32] A. V. Kumar and A. Wood. Representation and design of heterogeneous components. In Proceedings of the Solid Freeform Fabrication Symposium, pages 179-186. University of Texas at Austin, August 1999.
[33] V. Kumar, D. Burns, D. Dutta, and C. Hoffmann. A framework for object modeling. Computer-Aided Design, 31(9):541-556, 1999.
[34] V. Kumar and D. Dutta. An approach to modeling multi-material objects. In Proceedings of the 4th ACM Symposium on Solid Modeling, May 1997.
[35] V. Kumar and D. Dutta. An approach to modeling and representation of heterogeneous objects. Journal of Mechanical Design, 120:659-667, December 1998.
[36] B. Lichtenbelt, R.Crane, and S. Naqvi. Introduction to volume rendering. Prentice-Hall. 1998.
[37] William Lorensen and James Miller. Visualization Toolkit extreme testing: A production release every day. In Proceedings of the 13th international software/internet quality week. May-June 2000.
[38] M. Mäntylä. An introduction to solid modeling. Computer Science Press, 1988.
[39] A.J. Markworth, K.S. Ramesh, and W.P Parks. Modelling studies applied to functionally graded materials. Journal of Materials Science, 30:2183-2193, 1995.
[40] A.J. Markworth and J.H. Saunders. A model of structure optimization for a functionally graded material. Materials Letters, 22:103-107, January 1995.
[41] Materialise NV. Materialise software: Magics. [online] http://www.materialise.be/magics, 2000. Accessed Aug 2000.
[42] D. Meagher. Geometric modeling using octree encoding. Computer Graphics and Image Processing, 19(2):129-147, 1982.
[43] R. Mercado. Voxel modelling for rapid manufacturing. Master's thesis, Faculty of Computing Sciences and Engineering, De Montfort University, 1998.
[44] E. Miya. FAQ (Frequently Asked Questions) for the newsgroup comp.graphics.visualization [online]. Posted regularly on the newsgroups comp.graphics.visualization, July 1998. Accessed July 1998.
[45] Y. Miyamoto, W.A. Kaysser, B.H. Rabin, A. Kawasaki, and R.G. Ford. Functionally Graded Materials: Design, Processing and Applications. Kluwer Academic Publishers. 1999.
[46] M.J.Bailey. Tele-manufacturing: Rapid prototyping on the internet with automatic consistency-checking. IEEE Comput. Graph. Appl., 15(6), November 1995.
[47] S. Morvan and G. Fadel. Heterogeneous solids: possible representation schemes. In Proceedings of the Solid Freeform Fabrication Symposium, pages 187-197. University of Texas at Austin, August 1999.
[48] S. Muraki. Approximation and rendering of volume data using wavelet transforms. In A. Kaufman and G. Nielson, editors, Visualization '92, pages 21-28, Boston, MA, October 1992. IEEE Computer Society Press.
[49] S. Muraki. Volume data and wavelet transforms. IEEE Computer Graphics and Applications, 13(4):50-56, 1993.
[50] S. Muraki. Multiscale volume representation by a dog wavelet. IEEE Transactions on visualization and computer graphics, 13(4):50-56, June 1995.
[51] Advisory Group on Computer Graphics. Review of visualization systems. [online] Available at http://www.agocg.ac.uk/reports/visual/vissyst/vissyst.pdf, 1995. Access October 2000 .
[52] L. Piegl and W. Tiller. The NURBS book. Springer, 1997.
[53] F. Prinz et al. Rapid Prototyping in Europe and Japan. [online] http://itri.loyola.edu/rp/, March 1997. Accessed Aug 2000.
[54] B. H. Rabin and I. Shiota. Functionally gradient materials. MRS Bulletin, XX(1):14-15, January 1995.
[55] J. Rossignac and A. Requicha. Solid modeling. In J. Webster, editor, Encyclopedia of Electrical and Electronics Engineering. John Wiley and Sons., 1999.
[56] H. Samet. Region representation: Quatrees from binary arrays. Computer Graphics and Image Processing, 13:88-93, 1980.
[57] W. Schroeder, K. Martin, and B. Lorensen. The Visualization Toolkit: An ObjectOriented approach to 3-D Graphics. Prentice Hall, second edition, 1998.
[58] Schroff Development Corporation. Schroff development corporation. [online] http://www.sdcpro.com, August 2000. Accessed Aug 2000.
[59] B. Stroustrup. The $C++$ programming language. Addison Wesley, 2nd edition, 1991.
[60] M. Stytz et al. Three-dimensional medical imaging: algorithms and computer systems. ACM Computing Surveys, 23(4):421-499, 1991.
[61] W. Su. Fabrication of FGMs using laser powder fusion. Internal report Rapid Manufacturing Group, DMU, August 2000.
[62] R. Williams. Data powers of ten. [online] http://www.ccsf.caltech.edu/~roy/dataquan/, No date. Accessed October 2000.
[63] R. Winder and G. Roberts. Developing Java Software. John Wiley, 1997.
[64] W.Schroeder and K.Martin. The vtk User's Guide, June 1999. Kitware, Inc.. 1999.
[65] B. Yeo and B. Liu. Volume rendering of DCT-based compressed 3D scalar data. IEEE Trans. Visual. Comput. Graph., 1(1):29-43, March 1995.
[66] S. You et al. Volume rendering for virtual colonoscopy. In Proceedings IEEE Visualization '97, pages 433-436, 1997.

## Appendix E

## Glossary

ABS Acrylonitrile Butadiene Styrene. ABS is thermoplastic and can be easily heat shaped.
B-Rep Boundary Representation.

CAGD Computer Aided Geometric Design.

CFD Computational Fluid Dynamics.

CSG Constructive Solid Geometry

CVD Chemical Vapor Deposition. A method of creation of FGMs by deposition of gases at high temperature on a substrate. This method has been used to create SiC depositions on a C substrate and zirconium carbide/carbon ( $\mathrm{Zr} / \mathrm{C}$ ) depositions on a $\mathrm{C} / \mathrm{C}$ composite. The composition is controlled by varying the source gas mixture.

CVI Chemical Vapor Infiltration.

FEA Finite Element Analysis.

FEM Finite Element Method.

FGM Functionally Graded Material.

IGES Initial Graphics Exchange Specification.

LENS Laser Engineering Net Shaping (Optomec Design Co.) A rapid tooling method to create fully dense metal parts.

LOM Laminated Object Manufacturing. Laminated Manufacturing method commercialised by Helysis. It is described in section 2.5. works by stacking sheets of material and cutting the outline of every layer with a laser.

NURBS Non-uniform Rational B-Splines.

OMT Object Modeling Technique

RP Rapid Prototyping.

RT Rapid Tooling.
r-set A mathematical representation of a solid based on point set theory.

SDC Schroff Development Corporation.

SFF Solid Freeform Fabrication. The preferred term to refer to Rapid Prototyping in the USA.

STEP Standard for the Exchange of Product Model Data. A comprehensive ISO standard (ISO 10303) that describes how to represent and exchange digital product information.

STL Stereolithography file format. This is the format developed by 3-D Systems and the industry de facto standard format for object information transfer to RP machines.

TBC Thermal Barrier Coating. A successful application of FGMs for the thermal protection of components.

VTK The Visualization Toolkit, an open source, freely available software system for 3-D computer graphics, image processing and visualisation.


[^0]:    ${ }^{1}$ Acrylonitrile Butadiene Styrene

[^1]:    ${ }^{1}$ The notation used in the algorithms represents assignment with the left arrow symbol $(\leftarrow)$ and $\mathrm{C}++$ style indirections from a pointer to a structure to a member with the right arrow symbol $(\rightarrow)$

[^2]:    ${ }^{1}$ The notation used in the algorithms represents assignment with the left arrow symbol $(\leftarrow)$ and $\mathrm{C}++$ style indirections from a pointer to a structure to a member with the right arrow symbol $(\rightarrow)$ as noted previously on page 141 .

[^3]:    ${ }^{1}$ PC, Windows NT 4, 128 MB RAM Pentium III Processor at 400 MHz

