Aspects of the automation of casting pattern making

Phelan, Nigel R.

Award date:
1992

Awarding institution:
University of Bath

Aspects
of the
Automation
of
Casting Pattern Making

submitted by

Nigel R Phelan

for the degree of PhD

of the

University of Bath

1992

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Acknowledgements

I would like to express my gratitude to a number of people for their help and encouragement during the course of this project. Firstly to my supervisor, Adrian Bowyer and my colleagues Andrew Wallis and Ian Walker. I would also like to thank James Davenport and Bernard Silverman and numerous other members of the School of Mathematical Sciences for encouraging me to keep working on the write-up.

Dedication

This thesis is dedicated to Karen, without whose patience and gentle nudging I would never have got around to finishing it.

Abstract

Pattern making for casting is still largely a craft practiced by skilled pattern-makers. As two different pattern makers will not create identical patterns for the same casting, problems can arise in downstream fettling and machining processes, particularly if these are automatic. Also, variations in castings from one pattern to another are greater than those between castings off the same pattern - the casting process is more repeatable than the pattern-making process.

This thesis examines a number of new methods for pattern making using geometric modelling. The ultimate aim of the work of which this thesis is a part is to allow a pattern to be generated completely automatically from a geometric model of the casting which is to be made. To this end, a novel algorithm for automatically determining the split of a casting into cope and drag patterns, and - if needed - also cores is described. In addition, methods for automatically tapering castings have been developed, and an interactive program has been written to allow the gate, runner, and riser system to be easily added to patterns.

The work was sponsored by Lister-Petter plc. Examples are given in the thesis of the algorithms it describes working on both simple geometrical test shapes, and on geometric models of real Diesel engine components such as cylinder heads.

Contents

List of Figures

Chapter 1

Introduction

1.1 Purposes of the Project

This thesis describes the work carried out by the author between October 1986 and September 1989 with the assistance of funding from the SERC and Lister Petter PLC of Dursley. The project for which this funding was provided was concerned with the development of algorithms to allow computers to be used to assist in the process of casting pattern design and manufacture.

The areas of particular interest to the author were:

- The development of algorithms to automatically apply taper to a proposed casting design
- Investigating the problems associated with the automatic selection of suitable locations at which to separate a casting design into regions for which a usable pattern can be manufactured.
- The development of tools to assist the process of gating design.

There are a number of specific problems which require attention in these areas. Tapering algorithms should ideally be applicable to a wide range of pattern

shapes, and, as far as possible, consistent in their handling of similarly-shaped components. They should also be able to handle the full range of surface geometries. The automatic detection of suitable directions for the separation of pattern elements from the mould requires some form of interference checking, to ensure that the pattern does not damage the newly produced mould as it is extracted, and a mechanism for selecting optimal separation directions which allow the pattern and mould to be composed of the smallest number of separate elements. Little progress has been made, particularly in the latter area, by previous research.

1.2 General Advantages of Automation

There are a number of difficulties that arise when the pattern for a casting is designed and made in a conventional pattern shop. Amongst them are the following:

- If several patterns are produced for the same component there is often more statistical variation between components produced from different patterns than there is for components produced from one pattern on different production runs. In other words, there is less repeatability in the pattern creation process than there is in the mould manufacturing and metal pouring processes.

- It is difficult to perform the necessary double inversion of regions of solid and of air to translate from a normally 2D drawing of a complex casting to the 3D pattern set that will produce it. Even a skilled pattern maker will often need to modify a pattern set based upon the results of test pourings.

- The fact that the positioning of runners, risers and split lines on a casting is normally left to the discretion of the pattern maker means that two pattern sets for the same component may produce castings which differ subtly from one another. This makes it difficult to automate subsequent stages, such as

fettling and machining.

- The only way to record the pattern is physically to store it on a shelf. This takes up a lot of space.
- It takes a long time to produce a pattern given the casting design that it is required to produce.

1.3 Collaborating Bodies

1.3.1 Lister Petter PLC

The diesel engine manufacturing company Lister Petter PLC, was formed in 1985 by the merger of R A Lister & Sons Ltd, and Petter diesels. It has a world-wide reputation for the manufacture of stationary engines and exports a large part of its output to Europe and the Middle East; investing heavily in new technology, it is a DTI demonstration site for Computer Integrated Manufacturing (CIM) techniques. Its software division, COTEC, provides computing and consultancy services both in-house and to a number of other companies. Lister Petter PLC is a division of the Hawker Siddley Group.

Diesel engines are constructed from a number of cast components, as is the case for most internal combustion engines. As a result, Listers have a foundry on site, which provides the supply of castings for their engine manufacturing process. Their pattern shop is currently one of the few parts of their manufacturing facility which is handled entirely manually.

1.3.2 Solid Modelling Group, School of Mechanical Engineering, University of Bath

The School of Mechanical Engineering has an active research interest in the field of solid modelling, which was pioneered by Dr John Woodwark[1], who joined the school in 1978. Under his guidance, a number of solid modelling research projects were initiated, looking at efficient algorithms for representing engineering components. Several modelling systems have been written within the group, all based on a common philosophy: attempting to minimise the degradation of performance incurred by the system in response to increasing complexity in the objects being manipulated. The work described in the following chapters follows this same philosophy.

1.4 Overall Structure

This thesis discusses several aspects of the problems associated with attempting to automate the patternmaking process, and presents possible approaches to several of them. Chapter 2 provides an overview of patternmaking practices and terminology, and of the mechanisms used by geometric modelling systems. Chapter 3 surveys previous work on the application of computerised techniques to foundry processes, and problems which resemble aspects of the patternmaking process. Chapter 4 addresses the problem of applying taper to a pattern in an automatic manner and presents some conclusions about which approaches appear to be most promising. The problem of selection of appropriate split lines for a casting is considered in chapter 5, which also discusses the details of a mechanism devised by the author for performing this process. Chapter 6 presents a discussion of a graph-

[1]Dr Woodwark left the school in 1986 to become Graphics Manager at IBM's UK Scientific Centre, in Winchester. He is now editor of the journal "Computer Aided Design", and a director of Information Geometers Ltd., a geometric modelling consultancy.

ical interface implemented by the author which allows for the design of a gating system for casting designs based upon a planar split line, run on the joint. Chapter 7 summarises the author's conclusions about automating the patternmaking process, and Chapter 8 presents recommendations for further work in this area.

Chapter 2

Description of the Problem

2.1 An Outline of the Casting Process

This work described in this thesis is based on the assumption that one is considering the production of cast metal objects in sand moulds, although much of the discussion applies to other types of casting as well. This section is a summary of the process, based largely on the author's observations, and the advice given, during a two month induction course in the Lister-Petter foundry and pattern shop at Dursley, which he undertook at the outset of the project.

Sand Casting resembles two other common production techniques: die-casting and injection moulding. In all of these processes molten material is put into a suitably shaped cavity and then cooled until it solidifies. There is, however, one important distinction between sand casting and these other processes: the cavity, into which the molten material is poured, is in a cheap, fragile medium, and the finished casting is removed from the mould by breaking up the mould. This allows far more intricate shapes to be produced using this technique, since one is not required to be able to put the mould back together after removing the casting from it. Die casting and injection moulding are generally used for simpler component geometries. They use durable moulds, normally incorporating

some mechanism for automatically opening and closing the mould cavity, a cooling system, and some device for ejecting the finished part from the cavity. This allows them to produce large numbers of simple components rapidly.

The casting process starts with a *pattern*. This is used to form a cavity in the moulding medium, either by pressing the pattern into the material or by packing material around it. The pattern is then withdrawn from the newly formed mould, leaving a cavity into which molten metal will be poured. It is this need to remove the pattern from the mould that introduces one of the main complications into the process of pattern design – the need for multi-part patterns. For almost all castings, if one simply packs the moulding medium around a suitably shaped pattern, it will be impossible to remove the pattern subsequently without disturbing the shape of the mould cavity. This is not a problem in *investment casting*, where the pattern is either vapourised by the heat of incoming metal as the casting is poured, or melted out prior to pouring, but in processes which seek to re-use the pattern another approach must be adopted. The normal solution is to make the mould in several pieces, using one pattern element to form each part, and then to assemble these pieces. This means that the pattern maker has to choose suitable *split lines* along which to separate the pattern into different parts. The pattern maker may also choose to make the mould in multiple parts, to reduce the complexity of the pattern required for each part to manageable level. In this case a *pattern set* will be needed for each part of the mould assembly.

2.1.1 The Pattern Making Process

This section describes the process applied to a new casting design in order to produce a complete set of patterns for it.

All molten metals undergo some volume reduction as they solidify and cool. The amount of shrinkage experienced varies from metal to metal but is of the order of 0.5–1.5% on linear dimensions. To compensate for this it is necessary to

scale up the dimensions of the part by a factor appropriate to the material being used.

It is also necessary to make an allowance on the casting for surfaces which will subsequently require machining. When working with cast iron, such surfaces are normally built up by up to 3mm (1/8"), or twice this on large[1] hole diameters[2].

At this stage, it is important to decide how many parts the mould is to be made in. It is rarely possible to produce a single piece mould, except for very simple components where there is at least one surface where the standard of finish is not critical, eg iron billets. Thus it is normally necessary to make a decision about where the mould is to be split. This decision is influenced by the shape of the part, since there will normally be only a limited range of directions in which parts of the pattern can be withdrawn from the mould, and these separation directions will only be consistent with certain split lines.

For ease of manufacture and use of the pattern set, it is desirable to use planar split lines whenever possible. For many components with complicated shapes, however (notably engine manifolds), no suitable single plane can be found. The number of separate parts of the mould should be kept to a minimum, but regions which are undercut, or which have internal detail, will require more complex solutions, such as the use of cores, or loose pieces on the pattern, or drawbacks (see Page 21).

Having decided upon the split line or lines for the mould, it is possible to start to design the pattern equipment. This basically consists of inverting the geometry of each element of the mould, so that each element which was solid becomes air and vice-versa.

Once this is done, taper must be applied wherever the axis of separation lies

[1]Small holes are normally omitted from the casting and added by subsequent machining operations.

[2]The double allowance on diameters is, of course, equivalent to making the same allowance on the hole radius.

in the plane of a surface. Taper, or *draft*, is used to prevent surfaces from sliding past one another when the mould is separated from the pattern. It is imposed by deflecting such surfaces away from the line of separation by a small angle, in the range of 0.5 to 12 degrees, depending upon the size and complexity of the feature to which it is being applied. Taper reduces the amount of force which must be applied to the pattern in order to separate it from the mould.

Any sharp corners within the pattern should be rounded off: sharp edges generated in the mould will have a tendency to break off and wash away during casting, and sharp edges on the casting will give rise to stress concentrations and the metal will tend to draw back during cooling and solidification.

Mould yield is important, so smaller items are usually produced with a pattern that makes several castings at once.

For small to medium sized castings, some form of *plate work* is often used. The pattern is made up and attached to a backing plate. This is then pressed into the mould, automatically forming a planar split line as it creates the mould cavity. This process is therefore applicable only to parts which can be formed using a planar split line. There are several variations on this basic idea:

The pattern may be made up in two halves, these being fitted on opposite sides of the plate. One half of this is then used to form the top part of the mould, or *cope* and the other to form the bottom, or *drag*. This arrangement is known as a *match plate*. The pattern may be made up on two separate plates, one for the cope and one for the drag. This method is used for a number of conventional moulding techniques.

Moulds

Open sand moulds are used for low grade castings such as fire grates. Their top surface is exposed to air, and can thus only be used where the component includes a flat, non-critical surface. Where better quality castings are required, it

is necessary to use a closed mould, where the casting is completely surrounded by sand.

Green sand is a widely used moulding medium. It consists of clay and water-bonded sand, which is compacted around the pattern and which retains its shape largely as a result of how firmly it has been compacted. As a result, it is not well suited to the moulding of delicate features. If this is necessary some form of resin binder may be employed, eg cold set (using cold curing resin) or shell moulding (using thermosetting resin). The latter process is particularly suited to reproducing fine detail, but the materials are expensive. To try to prevent the molten metal from washing away sand and burning through the resin, the mould is normally coated with a refractory material.

Cores (see later) are often *blown*: resin coated sand is blown, normally using nitrogen gas, into a core box and cured by heat or a catalyst. This approach requires that the core box have vents in it to allow the gas to escape whilst retaining the sand.

The sand is often reinforced with metal pins known as *sprigs*, *wires* or *moulder's brads*. These help the mould and cores to resist the up-thrust forces exerted by the molten metal on the less dense sand, which can be quite considerable.

In order to form a strong mould, it is necessary to ram the sand firmly. There are a number of methods for achieving this. Small moulds, where a cold-setting resin binder is being used, may be rammed down by hand. Where a greater ramming pressure is required, a hand operated ramming machine may be used. For green sand work, where high ramming pressures are required, a moulding machine is often used. There are several types of these:

A *squeeze moulding machine* compacts the mould by pressing sand down onto the pattern. This produces a mould which is firmly packed at the back and less so around the pattern surface.

A *jolt moulding machine* uses a pattern mounted on a table which is in turn

mounted on a ram. The mould is filled and then the table is made to drop repeatedly against a stop. This produces a mould in which the sand around the pattern is firmly packed whilst that at the back of the mould is less so.

A *jolt-squeeze machine* uses a combination of the above techniques, the mould being first jolted and then squeezed. This gives a fairly uniform packing density.

A *sand slinger* works by throwing scoops of sand into the mould, the force of impact serving to ensure that the mould is well packed. It must be used carefully, since sand must not be allowed to build up in one place if good results are to be obtained.

A *blow-squeeze machine* blows sand into the mould cavity and then squeezes it to form a very stable mould. There are a number of variations on this technique.

These machines may also incorporate other refinements to assist the moulding process. Most large machines have a mechanism for removing the pattern from the mould (pattern draw). Many will also produce both cope and drag, using different patterns, and, after drawing the patterns, will turn the cope over and place it on top of the drag (roll-over). Large pattern plates may have electrical heaters to reduce the risk of sand sticking to them during the drawing, and the pattern draw mechanism may incorporate vibrators, again to assist with the drawing process.

2.2 Aspects of the Pattern Making Problem

2.2.1 Splitting

Undercuts and the need for Split Lines

Even after selecting split lines with care, there will normally still be a number of features on the finished part which, if included in a simple pattern, would prevent the pattern from being removed from the mould once the desired shape had been formed. These features correspond to areas on the finished part which

are undercut, or which have internal detail. It is to deal with problem areas such as these that cores, loose pieces and drawbacks are used.

Cores, Loose Pieces and Drawbacks

Cores are loose parts of the mould which are made up separately in *core boxes* and are then inserted into the mould to form the features which are causing the difficulty. They have a number of locating lugs moulded into them. These are designed to fit into corresponding features on the mould, to ensure that the positional accuracy of the features is maintained. Such a combination of a locating lug and a hole to accommodate it is known as a *core print.* With careful use of cores, it is possible to produce very complex forms, but it is necessary to design a new core box for each core, which can greatly increase the time taken to prepare a set of patterns, their cost, and also the amount of time taken to produce a complete mould.

If the feature which is causing difficulty is an undercut, it may be possible to avoid the expense of a separate core box by using a *loose piece* in the pattern. If the features concealed by the overhang are formed by a part of the pattern which can be left behind when the bulk of the pattern is withdrawn, it may be possible to ease out this loose piece separately, in a different direction to the rest of the pattern. This technique is clearly best suited to situations where moulds are made up by hand, eg cold set, and is of more limited scope than the use of cores.

A third method, which is sometimes used to allow the creation of undercut details, is the use of a *draw back.* This is a part of the mould which is so supported that it can be built up on top of the pattern and then drawn back to one side of it to allow the pattern to be lifted out. This technique is more common on larger jobbing castings and cannot be used in automatic moulding plants or core machines.

2.2.2 Tapering

Taper is applied to castings in order to simplify the mould making process. After the moulding medium has been packed around the pattern, it is necessary to withdraw the pattern in some direction to form a cavity into which metal may be poured. If, however, there are surfaces within the mould whose surface normals are perpendicular to the direction of withdrawal of the pattern, parts of the pattern will slide past them as the pattern is removed, possibly dragging material away from the surface of the mould and thereby damaging it.

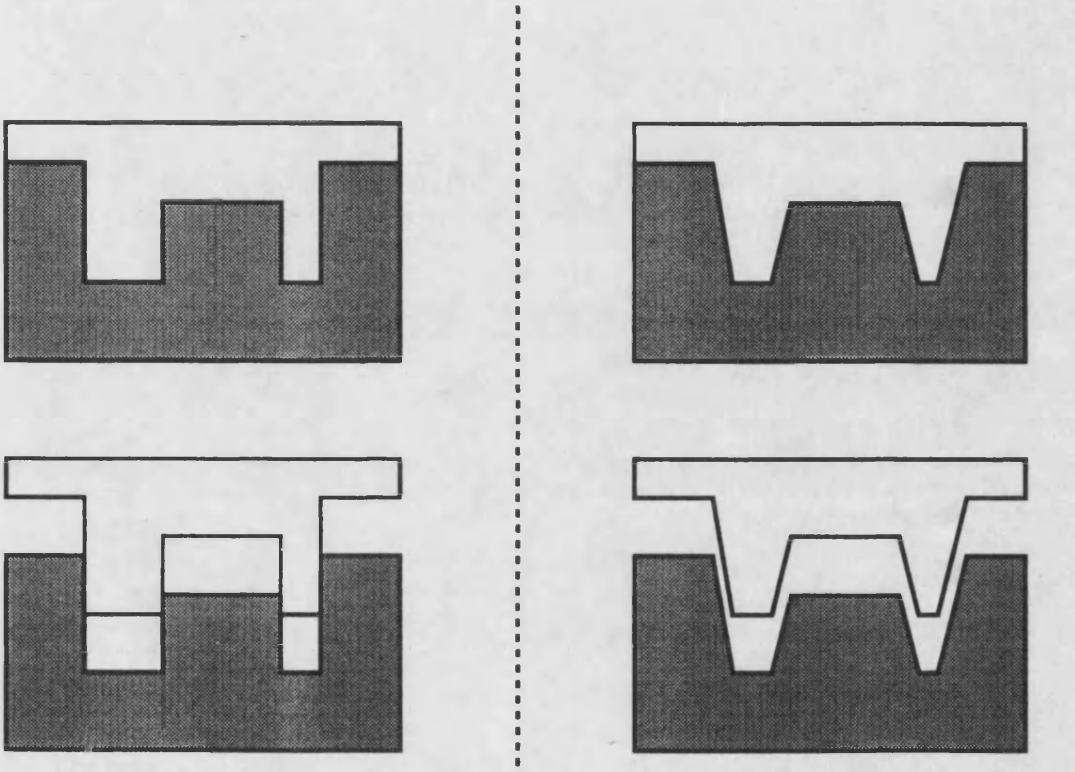
If the surfaces whose normals were perpendicular to the direction of pattern withdrawal are now inclined slightly, the risk of mould damage is greatly reduced. In this case, the pattern ceases to be in contact with the mould after a small displacement, diminishing the opportunities for damage to occur (See Fig 2-1).

2.2.3 Design of Gates and Risers

Gating systems are used to supply molten metal to all parts of the mould cavity. They consist of a series of channels cut into the mould joining the pouring cup to the *ingates* where the metal enters the mould cavity itself. Careful design of the gating system is important to the production of successful castings.

It is usually a good idea to arrange for the molten metal to flow into the mould cavity through what will be the thinner sections of the casting. This means that the thinner sections, which tend to cool more rapidly, will receive hotter metal to compensate. The gating system must be large enough to allow for an adequate flow of metal into the cavity, otherwise it will not fill properly, giving rise to a defective casting. It will save time at the fettling[3] stage if the runner system is designed to break off easily, and it can reduce scrap if the areas where the ingates

[3]Fettling is the process of removing *flash* – the thin fins of metal that are normally found on the parting lines of a casting, where minor misallignments have allowed molten metal to seep between mould elements, and also the lumps left where the gating system was formerly attached



No Taper - friction on vertical faces during pattern removal

With Taper - pattern separates from mould after a small displacement

Figure 2-1: The use of taper

attach are built up, thereby reducing the risk of damage to the casting when the ingates are broken off.

There are three basic runner system layout philosophies. They are known as *top run*, *run on the joint* and *bottom run*.

In a *top run* gating system the metal is poured straight into the top of the mould cavity. This gives rise to highly turbulent flow patterns, so there is a significant risk of mould erosion and of sand and slag inclusions occurring within the casting. There are a number of refinements to this basic technique, intended to reduce this risk, but this feeding arrangement is not normally used for high-quality castings.

A gating system which is *run on the joint* feeds the metal in through a series of ingates along the split line of the mould. It is probably the most popular from of gating, since it gives reasonably good results and it is much easier to design this type of gating into a pattern.

Where it is very important that castings be clean, a *bottom run* gating system may be used. Here the molten metal is fed around to the bottom of the casting before it enters the mould cavity. The metal is able to rise smoothly up through the cavity without any risk of washing away sand and thus gives more consistently good castings, at the expense of a more complicated mould design.

2.2.4 Problems Associated with Shrinkage

There are two basic types of shrinkage: *microshrinkage* and *macroshrinkage*.

Microshrinkage occurs at the level of the metal's crystalline structure. As metal crystals solidify out of the melt, liquid metal is trapped in the spaces between them. This cools, solidifies and contracts, giving rise to flaws with a visible crystalline structure. This kind of shrinkage is normally found in alloys with a large interval between the temperature at which solidification commences and that at which it ends.

Macroshrinkage, where large voids occur within a casting or surfaces shrink in on themselves, is a characteristic feature of the part of a casting which cools down most slowly due to the presence of a local heat centre. This kind of shrinkage can be minimised by controlling the solidification and cooling processes and forcing the shrinkage to occur outside the casting in a *feeder*.

A *feeder* is a reservoir of metal attached to the casting in such a way that it can supply molten metal to the casting as this contracts during solidification. The feeder is intended to feed the hottest part of the casting, which will be the last region to solidify. Thus it must be fairly close to this part and must have a longer solidification time than it. This latter point means that its shape must offer a large heat capacity for a given volume and allow a good flow of metal into the casting. It must obviously be large enough to hold a sufficient volume of metal to compensate for the effects of casting contraction. The *neck*, connecting the feeder to the casting, must also be designed carefully since it must remain open, ie must not solidify, whilst the casting is setting.

In order to design a suitable feeder, it is necessary to locate the part of the casting which will take the longest time to cool. This can be done by considering the most compact parts of the casting and calculating their moduli of solidification. The modulus of solidification of an object is the ratio of its volume to its cooling surface areas. In practice, each element of the casting is normally approximated to a simple geometric form and the correct formula for that shape applied. Those parts of the casting which have a large modulus of solidification are likely to be the parts which cool most slowly, and thus which are most likely to require some form of feeding. A technique known as *directional solidification* attempts to avoid the need for feeders by stipulating that the modulii of parts of the casting should decrease as one moves away from the ingates from which the casting is fed. This method is inaccurate, since it neglects many factors, such as the cooling effects of fins close to a hot spot and the isolating effects of an adjacent core, but it is

widely accepted as sound casting design practice.

For further discussion of the casting and pattern-making process, the reader is referred to the books by Jain, Ekey and Winter, and Flinn [27, 16, 17].

2.3 Solid Modelling Techniques

Solid modelling is the process of representing three dimensional objects unambiguously within a computer. This is achieved by using an object representation which has the special property that one can perform *membership tests* on it. A membership test is a test which, given a point **P** and an object **O**, will return a boolean true or false value indicating whether point **P** lies inside object **O**. There are two solid modelling techniques in widespread use today; set theoretic modelling and boundary representation modelling.

2.3.1 Boundary Modelling

A boundary model stores an object as a series of lists: of the points making up each face, of the faces making up each surface, and of the surfaces making up the component. If the object is, in fact, an assembly, then there will be further lists: of the components making up each separate unit in the assembly and of the units in the assembly.

With näive forms of this representation, it is possible to describe objects which cannot exist. For example, Fig 2-2 shows a polygon with a 'hanging face'.

This entity can clearly be represented by such a method, but equally clearly it is not a real object. A true boundary modeller is required to perform some consistency checks in order to ensure that it does not categorise such an object as a solid.

One strategy to try to prevent the creation of this sort of object is to check the topological consistency of the model ie, to ensure that there are no extra or

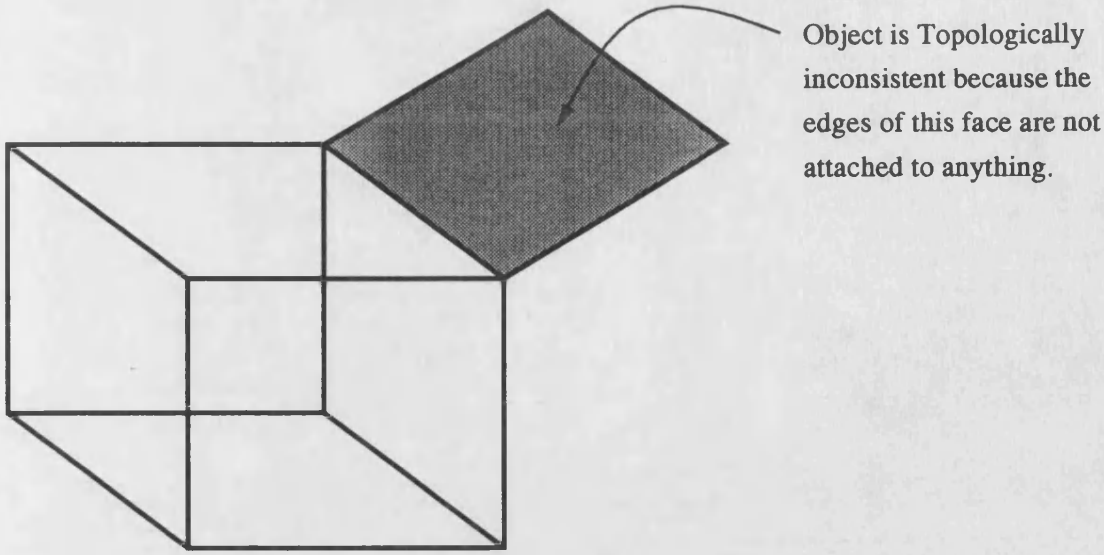


Figure 2-2: A *hanging face* in a boundary model

omitted faces, edges or vertices in the model. This can be ensured by applying Euler's rule: for a polyhedron with no through holes, this states that the number of edges must always be two less than the sum of the numbers of faces and vertices, ie

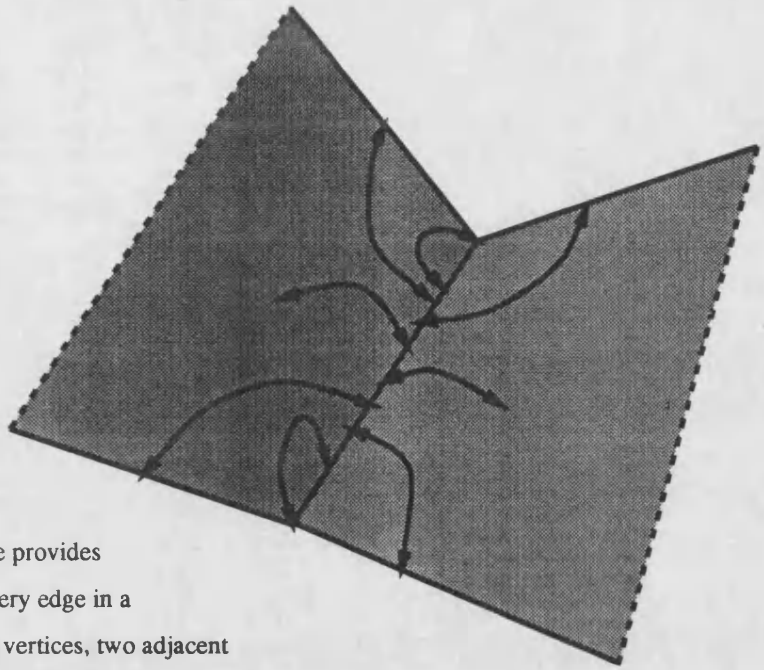
$$F + V = E + 2$$

where F is the number of faces, V is the number of vertices and E is the number of edges. Considering, for example, the case of a cube (6 faces, 8 vertices and 12 edges):

$$6 + 8 = 12 + 2$$

Extensions to Euler's rule can be applied to handle cases where through holes are present. One can also perform a number of simple tests, such as ensuring that each edge in the model lies between exactly two faces.

The data structure used to represent a boundary model is important to its efficient use; normally a large number of pointers are used in the representation



The winged edge data structure provides bi-directional pointers from every edge in a boundary model to its two end vertices, two adjacent faces, and the four other edges that share a face with it.

Figure 2-3: The *winged edge* data structure

of each entity to indicate adjacent elements. This allows a program to traverse the surface of the object without having to perform any list-searching operations. The so called *winged edge* data structure devised by Baumgart[2] has become something of a standard for boundary modellers. It is based upon a set of bi-directional pointers between each edge and its adjacent faces, vertices and edges (see Fig 2-3).

One problem with boundary models is that it is not sufficient to ensure topological consistency. Fig 2-4 shows an example of an object which satisfies Euler's rule but which is still not a real solid. It is, in fact, very difficult to verify that a given boundary model is actually a valid solid. The normal method for accomplishing this is to constrain the input system used by the modeller in such a way that it is incapable of producing invalid objects. This can be hard to achieve with a system that provides the designer with a sufficiently rich palette of functions to allow for the construction of complex objects in an intuitive fashion. In particular

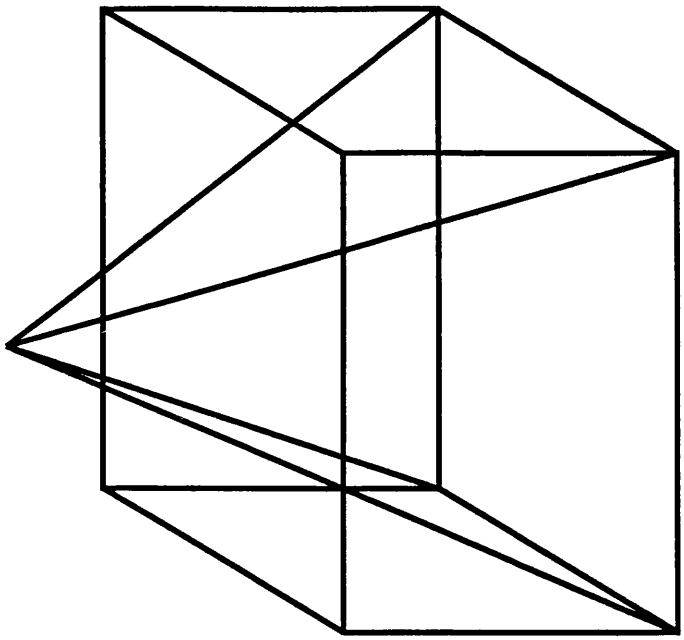


Figure 2-4: A topologically consistent *non-object*

the difference operator (subtract one object from another) can cause problems due to the finite numerical accuracy of the underlying computer. In the cases where the system allows the designer to 'tweak' a model by relocating arbitrary vertices then it is hard to guarantee that the resulting model will be a valid object.

2.3.2 Set Theoretic Representation

Set theoretic modellers represent an object as a collection of primitive elements. These are usually geometrically simple entities like boxes, cylinders, cones and tori. These primitives are joined together using boolean operators, usually union, intersection and difference (Fig 2-5). The model representation consists of a list of the primitives and the operators which must be applied to them to produce the complete object. This representation technique is sometimes known as Constructive Solid Geometry (CSG). This is a very natural technique for people to use to describe three-dimensional structures, and so most solid modelling systems use a set theoretic input method even when the internal representation method is

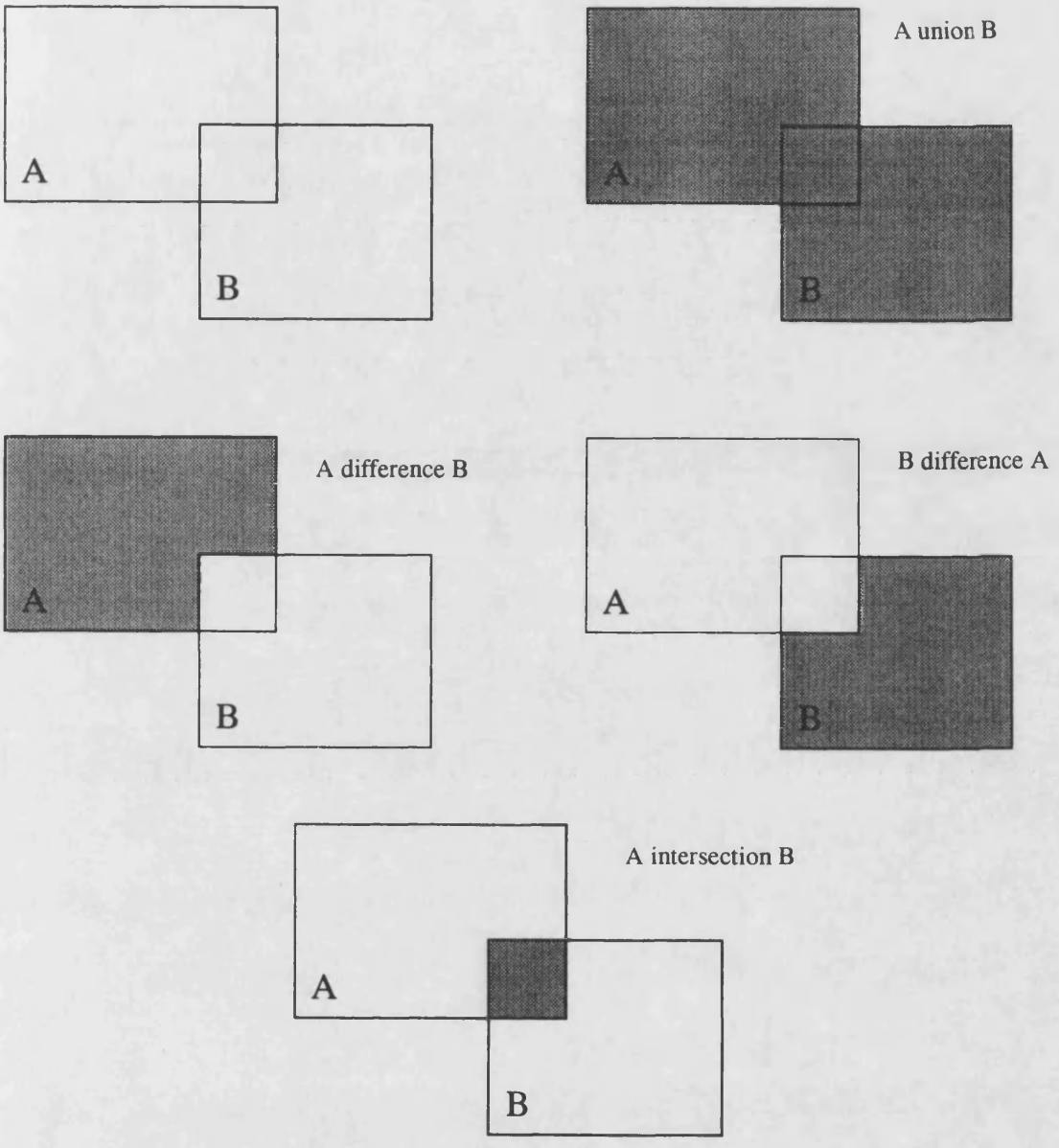


Figure 2-5: Set theoretic operators

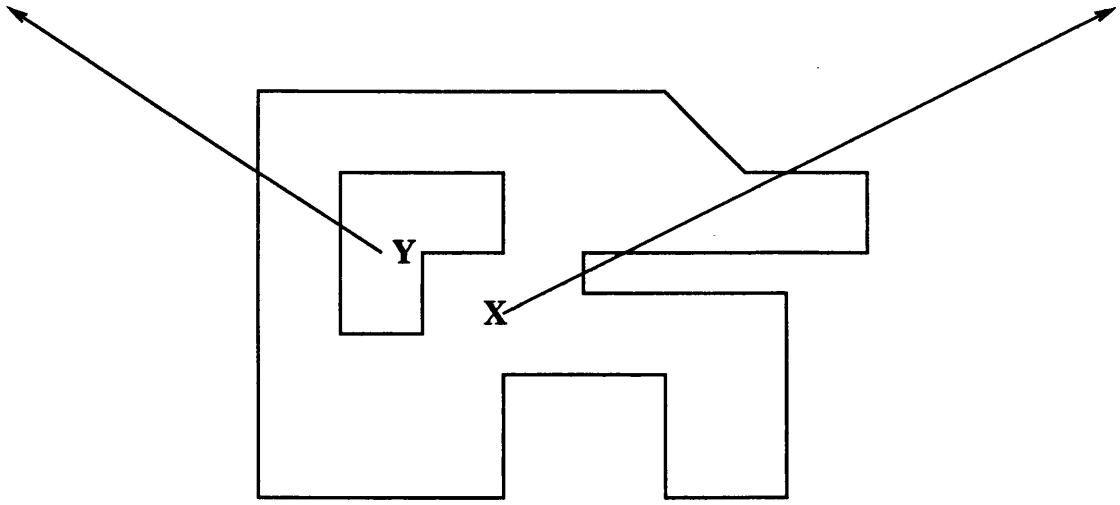
a boundary model.

Interrogating Solid Models

In order to perform a membership test on a boundary model, it is necessary to perform a ray-casting operation: a line is constructed from the point to infinity in some random direction. This line is then tested against every face in the model to see if it intersects it. Numerical accuracy problems arise if the line passes through or very near an edge or vertex on one of the faces and, if this occurs it will be necessary to generate a new test ray in a different direction and redo all the face intersection tests. If an even number of intersections is found, the point must be outside the object, whereas if an odd number is found it must be inside. This is illustrated in Fig 2-6.

With a set theoretic model a membership test is carried out in the following manner: the point in question is tested against each primitive in the object in turn. For each primitive there will be an algorithm which, given a point, will return a boolean value indicating whether the point is inside or outside the primitive. Once a result has been obtained for each primitive, the true/false results are substituted into the boolean expression describing the way in which the primitives are joined together. This expression is then simplified by application of the normal rules of logic to yield a true or false result indicating whether the point is inside or outside the object (See Fig 2-7). These operations are very computationally efficient, which means that set theoretic modellers are very good at handling this type of enquiry.

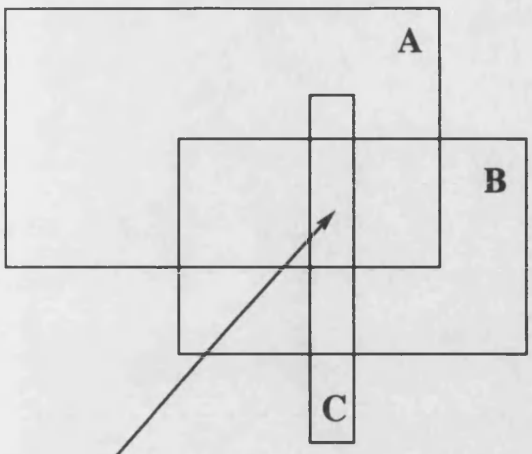
In order to obtain a picture of a set theoretic model one of two methods is employed. One can construct a list of the faces of the model, essentially by taking the face information for the primitives and modifying it appropriately to take into account the boolean operations applied to it, and then process this face information as one would for a boundary model (see later). Alternatively, one can

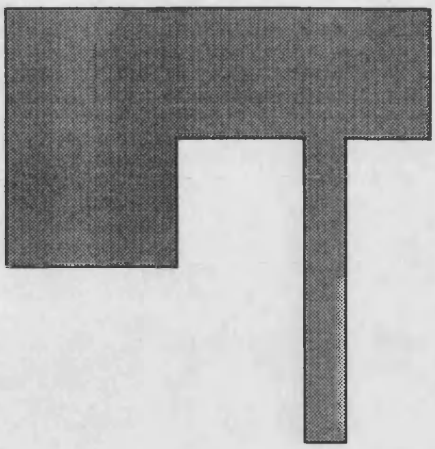


Project ray from point of interest to somewhere outside the model.
Compute number of intersections with surfaces of the model.
Even number implies point is outside the object.

So point X is inside, but point Y is outside.

Figure 2-6: Membership test on a boundary model





(A difference B) union C

Point X

X in A --> True

X in B --> True

X in C --> True

Substituting the results for X into the equation for the model:

(True difference True) union True

--> (False) union True

--> True

So point X is inside the object

Figure 2-7: Set theoretic membership test

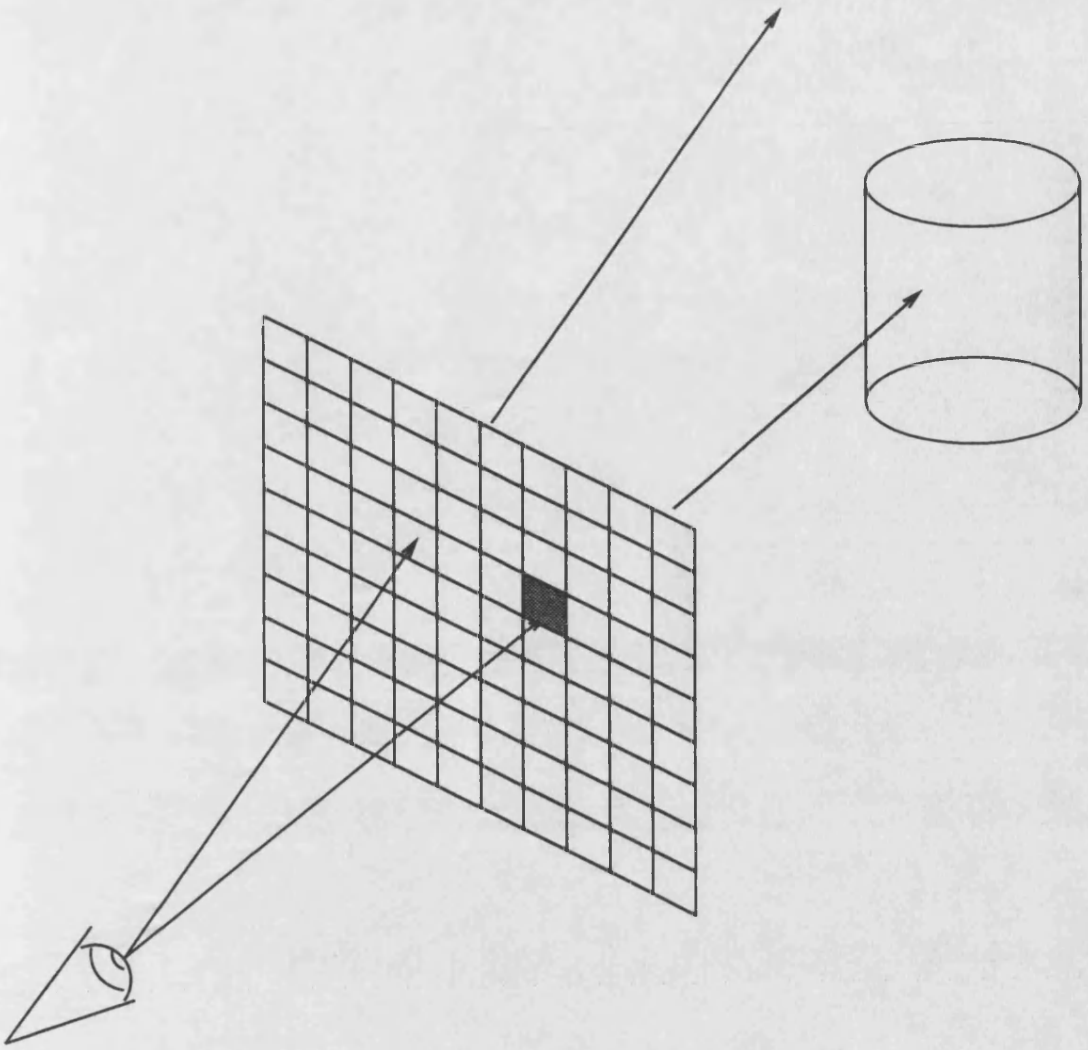


Figure 2-8: Ray Casting

employ a technique known as ray-casting. This is illustrated in Fig 2-8.

The object to be represented is considered to lie behind a plane in space where the two-dimensional output device (often a video display) is held to be. This output device will have some finite pixel resolution – say for example 1000 dots across by 500 dots down. For each pixel on the screen one computes the equation of a line from the point where the observer's eye is considered to be in front of the screen, passing through the pixel in question into the scene containing the object. Each primitive in the object is then compared with the equation of this

line to establish the points, if any, at which the line intersects the surface of the primitive. A list of all the intersections between the line and the primitives in the object is thus obtained. This list is sorted by distance from the observer's position. For each intersection, starting with the one closest to the observer, a membership test is performed. If the first point lies outside the object then the test is repeated for all subsequent points until a point is found which lies inside the object (actually it must lie on the surface of the object), or until all the points are exhausted. In the latter case the line does not actually intersect the object and the pixel on the screen which it corresponds to will be coloured according to some rule for handling the background scene. In the former case, however, the pixel is then coloured using the colour attributes associated with the surface on which the point lies, possibly taking into account the direction of illumination and other surface properties in order to enhance realism. By casting further rays from this point in appropriate directions and considering the objects which they intersect it is possible to simulate the effects of shadows and reflective, transparent or translucent surfaces. Repeating this process for every pixel on the screen will produce a picture of the scene containing the object which, by the addition of a few further refinements, can be made very realistic indeed. Much research activity is currently being devoted to the problem of achieving near-photographic realism in ray-cast images whilst attempting to minimise the time taken to generate them[54].

It is possible to generate pictures of boundary models by ray-casting, as for set-theoretic models, but, since surface information is directly available, faster rendering techniques are also possible. In particular it is trivial to generate a wire frame drawing. At the cost of increased computation time, one can improve upon this wire frame by eliminating from it lines and line segments which are obscured by parts of the model closer to the observer's view point. It is also possible to produce more realistic images by rendering the faces on the display.

Dedicated hardware now exists which will draw tens, or even hundreds of thousands of polygons per second, with automatic depth sorting of the polygon list to do hidden surface elimination and options such as Gouraud or Phong shading[43], which can be used to make a faceted surface appear to be smoothly curved. This hardware allows boundary modellers to display colour images of the objects being manipulated which can be rotated and altered with near instantaneous feedback to the designer. It is for this reason that most current commercial solid modellers are of the boundary modelling, rather than the set theoretic, type. It should, however, be noted that this is more a reflection of the previous directions of computer graphics research than any inherent advantage of the boundary modelling approach.

Other common requirements in a solid modeller are to compute the mass or volume of an object and to compute its surface area.

For an object of constant density, the problems of finding its mass or its volume are essentially equivalent. For a boundary model one can compute the volume of the object by sub-dividing the space it occupies into a number of sub-volumes, then performing a membership test within the sub-volume and adding it to the volume of the model if the test returns 'solid'. This method is, of course, equally applicable to a set-theoretic model. It should be noted that this is slightly more expensive in a boundary modeller because a membership test is a more expensive operation. On the other hand, a surface area calculation is cheaper in a boundary model, since face information is directly available and a summation operation is all that is required.

Another possible approach is to use a 'Monte Carlo' method. The system constructs a region in space of known volume, which is guaranteed to enclose the object. It then performs a series of membership tests on a randomly distributed set of points within this region. The volume of the object is estimated by multiplying the volume of the enclosing region by the ratio of the total number of points tested

to the number found to lie inside the object. The accuracy of the estimate can be increased to any desired level of confidence by increasing the number of points tested, at the cost of increased computational time.

Surface area may be calculated by generating a list of model faces as described above and then summing their areas, or by using a recursive spatial division technique to isolate regions of surface and approximate their area with polygons.

The differences between the two types of modelling system give them different strengths and weaknesses when it comes to manipulating the information which they contain. Since a boundary model explicitly contains point and surface data, it is easier to produce pictures of the object. On the other hand it is harder to compute mass properties since it is harder to test whether a given point is inside or outside an object. One of the features of a set-theoretic model is that it must always be a valid object (possibly not the one the designer intended, but certainly some real, three-dimensional entity). This is one of the great strengths of set theoretic systems when compared with boundary modellers.

In many ways, the ideal solution would be to be able to convert from one format to the other as necessary, using the most appropriate representation for any given problem. Unfortunately, whilst it is relatively easy to produce a boundary model from a set theoretic one, the reverse transformation is a difficult problem which has yet to be satisfactorily solved.

2.3.3 DODO and DORA

The set theoretic modelling systems which have been developed at Bath have been designed to use a slightly unusual, but very versatile, class of primitives — half spaces. A half space is an inequality in x, y and z which divides space into two regions: those points for which the inequality is satisfied, which are considered to be inside the object, and those for which it is not, which are held to be outside. A flat surface can, for example, be described by a planar half space of the form:

$$ax + by + cz + d \leq 0$$

The values of a, b, and c are normally constrained such that:

$$a^2 + b^2 + c^2 = 1$$

This describes a solid with a single surface, whose surface normal is (a, b, c) and which is a distance d from the origin at its closest point.

Two of the systems are still in regular use: DORA (Divided Object Raycast Algorithm) and DODO (Daughter of DORA). Of these, the DORA system is the earlier, and less complex, system. It handles only those objects which can be described entirely by planar half spaces. This does not, however, mean that it is incapable of handling curved objects: any curved surface may be approximated to any desired precision by the use of a suitably large number of planar facets. Use of this technique does, of course, mean that models of objects with curved surfaces in them will tend to contain large numbers of half spaces.

DODO was written by Dr Andrew Wallis whilst working at the University of Bath. It uses a number of the ideas upon which the DORA system is based, but is capable of handling a much richer class of surface primitives: any surface which may be represented as a polynomial in the space vectors x, y, and z can be used as the boundary of a primitive by the DODO system, which permits the exact representation of cylinders, cones and tori. The only common engineering surface which cannot be handled by DODO in its present form is the helix (used for screw threads), although a helical surface can, of course, be modelled to arbitrary precision by use of, for example, Taylor's series approximations to the transcendental functions (*sin*, *cos* and *tan*).

Unlike in most set-theoretic modelling systems, these primitives are not, in general, bounded: it is not always possible to draw a box of finite size which

completely encloses the primitive, since in principle, and often in practice, a polynomial half space may extend to infinity in some direction. This introduces additional complications into the algorithms used to process the model, but this is felt to be a reasonable price to pay for the added flexibility that it provides to users.

A common problem with set theoretic modellers is that the time taken to process a model increases rapidly as the number of primitives in the model increases.

In a näive implementation, the time taken to, for example, produce a picture of the object by ray-casting, tends to rise dramatically as the number of primitives in the scene under consideration increases, so that a system which produces pictures of a collection of spheres or cubes in a few seconds may take minutes or even hours to process a typical engineering component. The problem is essentially the need to perform a series of comparisons between primitives, which become more expensive as the number of primitives increase:

In order to decide on the colour to paint a given pixel, it is necessary to project a ray into the object space and then perform intersection tests between each primitive in the model and the ray, an $O(n)$ operation. Those points at which intersections do occur then need to be sorted into order moving away from the view point (say an $O(nlog(n))$ operation), and membership tests need to be performed at each of these points in order to establish where, if anywhere, the first intersection with a real surface within the model occurs (again, an $O(N)$ operation). This gives an overall order for the algorithm of n^2log^2n. If effects such as shading, reflection or translucency are to be incorporated, secondary rays must be cast from the point of intersection, and these must, in turn, be intersected with the primitives within the scene in the same manner as the original ray, possibly dramatically increasing the cost of the ray casting operation. The operation is so expensive because, at two points, it is necessary to perform tests against all the primitives in the model. If the number of tests which needed to be performed could be reduced in some way, then the speed of the ray casting operation could

be increased significantly for scenes involving large numbers of primitives.

The modellers developed at the University of Bath attempt to avoid the problem of performance decreasing rapidly as the number of primitives rises by spatially dividing the model, so that, when considering any part of the model, it is only necessary to take into account those primitives which are physically close to the area of interest. This tends, in practice, to give a near linear variation in computation time with model complexity[71]. This compensates for the fact that, since the primitives being manipulated are, at least potentially, unbounded, it is not possible to perform *boxing tests* — many solid modellers achieve significant performance improvements by maintaining lists of boxes in space which are known to enclose each of the primitives in an object. When performing, for example, an intersection test between two primitives, it is often more efficient to compare their bounding boxes first, since, if these do not intersect, the objects cannot possibly do so. This test is often much less expensive in computational terms than testing the equations which describe the primitives directly, so in situations where the majority of intersection tests return a negative result there is a clear advantage in using a boxing test.

Division

The process of dividing the box enclosing an object into a number of smaller regions is used by a number of set theoretic modelling systems. It reduces the effective complexity of the region of the model under consideration at any one time, by allowing features which are physically remote from the region of interest to be disregarded.

The most common implementation is oct-tree division: the box is divided recursively along the x, y and z axes to produce eight sub-boxes, all of the same size, and geometrically similar to the parent box. By using a suitable hierarchical numbering scheme, it is possible to compute the boxes adjacent to a given box in

a computationally efficient manner. This method does, however, tend to require a great deal of data storage for non-trivial problems. It also tends to produce an excessive amount of division: since division occurs simultaneously along the x, y and z axes, and always divides the current box into eight equal boxes, it is not possible to exploit information about the model being processed to place division planes where they will have the greatest effect. This can reduce performance significantly since one division more than the optimum value can introduce eight sub-boxes which must be taken into account when performing, for example, a ray-casting operation into the divided structure. Thus a bad or inappropriate division criterion can give very poor performance.

It is for the above reasons that the DODO and DORA systems do not use an oct-tree division strategy. Instead, a binary division scheme is employed. This gives similar results to oct-tree division, but is less prone to over-dividing. The choice of a suitable division criterion is a difficult problem – not least because it is a function of the purpose to which the divided model is to be put: a strategy for producing pictures from a particular view-point might favour a division strategy which avoids divisions perpendicular to the viewing direction, since the sub-regions thus produced will often both need to be interrogated. It is, however, worth introducing such divisions if a relatively simple feature in the foreground obscures much or all of a more complex feature in the background. On the other hand, if the model is to be interrogated in an essentially random fashion, an even division strategy is probably more appropriate. It is also necessary to bear in mind the relative costs of root-finding operations to perform ray–object intersection tests and of traversing the structure used to represent the divided model: it is pointless to divide the model further once the cost of evaluating a given region is comparable to the cost of traversing the data structure to locate the region in the first place. A more detailed discussion of this topic can be found in Wallis[60].

Pruning

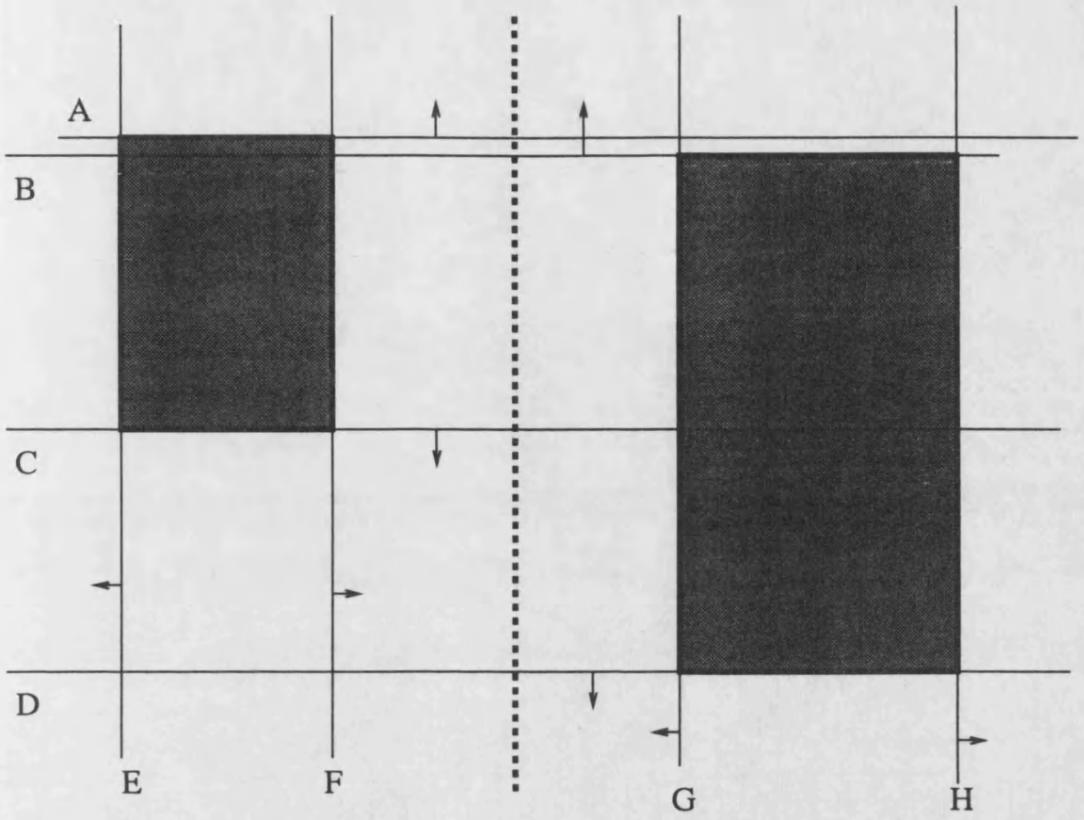
By itself, the process of dividing a solid model into a number of separate models corresponding to different spatial regions offers no performance advantages – it will, in fact, reduce the speed of model interrogation significantly, since it becomes necessary to find out which sub-box or boxes the query is directed towards, and to perform the interrogation on the model associated with that sub-box; indeed, with an interrogation such as a ray-casting operation, it may well be the case that several sub-boxes must be examined in turn in order to be able to return an answer to the query.

The division process only offers an advantage if it is coupled with a pruning process, which takes the model for a given sub-box and replaces it with a simplified model, which, within the confines of the box under consideration, is equivalent to the original. This is illustrated by Fig 2-9.

Clearly queries directed at the left hand side can now be answered more readily, since the object being manipulated is much simpler.

An important point to note here is that pruning is a more sophisticated process than simply the elimination of primitives which do not intersect the box under consideration: in Fig 2-9 half spaces B and D pass through the left hand region, but they are eliminated from the final expression because they are intersected with the half space G, which does not intersect the region in question. It is this property that makes pruning the powerful tool for controlling computational complexity that it is.

The process of pruning is carried out by considering the model of the object as a set of elements, joined by operators. Each of the elements is considered, in turn, with respect to the current sub-box. The question is then posed – *within this sub-box, can this element be replaced by a constant term (ie solid or air) in the model description without affecting the outcome of any query directed at the model?*. This may also be expressed as *does this element contribute only air,*



$$model = (A \cap C \cap E \cap F) \cup (B \cap D \cap G \cap H)$$

but if we concentrate on the left hand region:

$$G = air$$
$$H = solid$$

so we can simplify as follows:

$$model = (A \cap C \cap E \cap F) \cup (B \cap D \cap air \cap solid)$$
$$model = (A \cap C \cap E \cap F) \cup air$$
$$model = (A \cap C \cap E \cap F)$$

Figure 2-9: Pruning

or only solid to the model as a whole, throughout the current sub-box, or does it contribute a combination of the two; if the answer to this question is either *only contributes solid* or *only contributes air*, then the element is replaced by the corresponding term and the model is simplified accordingly by application of the rules of boolean logic.

There are four possible answers to the above queries:

- Element contributes only air to this sub-box
- Element contributes only solid to this sub-box
- Element contributes solid to some parts of this sub-box and air to others
- Element *may* contribute both solid and air to this sub-box

Most of these are self explanatory, but the fourth possibility merits some clarification.

It may be the case that the element under consideration is bounded by a polynomial half-space of relatively high degree. In such a case, the cost of determining whether the boundary actually passes through a given sub-box may be quite high. To avoid this cost, a technique such as interval arithmetic[38] may be employed to compute a box within which the boundary is known to lie, without locating it precisely. If this box is disjoint from the sub-box currently under consideration, or if it is wholly enclosed by it, it is clear how to classify the element we are considering (it must fall into one of the first three categories above). If, however, the two boxes overlap, or the bounding box encloses the sub-box under consideration, then one cannot state into which of the first three categories this situation falls, *ie* it is an example of the fourth situation. In order to maintain the consistency of the model, these cases must be treated as examples of the third category, *ie* they cannot be reduced to either solid or air.

It was previously stated that interval arithmetic can be used for this sort of classification. If, as in the case of the DORA modelling system, the class of

primitive objects is restricted to half planes, this is perfectly satisfactory, since interval arithmetic is exact for planar surfaces. As the degree of the surfaces bounding the primitive elements is increased, however, the bounds calculated by interval arithmetic grow rapidly wider. Techniques exist for compensating for this. One of the most interesting is that used by the DODO system – this allows arbitrary polynomial inequalities to be constructed from planar half spaces by 'multiplication' and 'addition' of planar half spaces, but retains information about the position of the half spaces from which the polynomial was constructed. This allows a more exact bounding box to be constructed than direct application of interval arithmetic to the polynomial inequality in x, y, and z would yield[59].

Alternatives to interval arithmetic are also in common use. A typical method is to provide a restricted class of primitives from which the object must be built. It is then fairly easy to provide a *does this primitive intersect this box* test for each type of primitive, particularly if the degree of the surfaces in the primitives is restricted. A more promising approach for the accurate modelling of arbitrary components is suggested by Milne[38], who uses a technique based on Sturm sequences to construct exact bounding boxes for arbitrary polynomial inequalities.

2.3.4 The Geometric Algebra System

Once a modelling system allows the use of arbitrary polynomial inequalities to represent primitives, and provides mechanisms for forming smooth fillet blends between them, the complexity of the equations which must be solved in order to, for example, render a picture of the object, increases dramatically, since a blend between two surfaces must, in general, be a surface of higher degree than either of those that it joins. This can lead to problems of instability or inaccuracy if numerical methods are used to solve them. In an attempt to deal with this problem, research is being carried out at Bath into the use of computer algebra

in solid modelling[4]. One of the results of this work has been the development of a Lisp–like language, *Whisper* [5], which is particularly well suited to the prototyping of algorithms for the manipulation of solid models. The software for computing casting split lines, described below, was originally implemented in this language using some of the features of the Bath Geometric Algebra System (GAS), which was also written in it.

2.4 Simulated Annealing

Simulated annealing is a global optimisation technique which is used to avoid the problem common in piecewise local optimisation techniques, of finding local, rather than global minima or maxima. This problem is inherent in any optimisation technique which works by trying to progress towards a goal by making small changes to an existing solution in order to try to find a better one.

The name for the process arises out of the analogy with the engineering process of annealing, whereby a metal is raised to an elevated temperature and cooled gradually to room temperature in order to relieve stresses within the material. It was first proposed by Kirkpatrick[33] and in essence it works as follows:

One starts from an initial solution, which is then evaluated using a cost function. One then makes a large number of random changes to it, and evaluates the result using the cost function. If this version scores better than the current 'best', it is adopted as the new 'best'. If it does not, one generates a random number and compares it with a threshold value. If the random number is less than the threshold value, the new solution is adopted as the current 'best', despite the fact that it scored less well than the previous solution. This process is then repeated a large number of times, with the number of random changes, and the threshold value for the random number test, being reduced repeatedly, until the threshold

[4]SERC/ACME grant ref GR/F 13171 *Computer Algebra and Solid Modelling*

value falls below some terminating limit.

It is the comparison between a random number and a steadily falling acceptance criterion that gives this algorithm its global optimisation properties – without this step, the process essentially reduces to the local optimisation technique known as *iterative improvement.*

From the above description it is clear that there are a number of constants which must be chosen with some care in order to achieve the best performance from this technique. If the threshold is reduced too slowly then it will take an excessively long time to obtain a solution; if, on the other hand, it is reduced too quickly there is a significant risk of becoming trapped in a local minimum or maximum and of thereby failing to find the best solution, or an approximation to it. This is an area where a certain amount of experimentation is called for, although some guidelines may be found in [18].

The technique tends to produce good, but non-optimal, solutions to a problem. The principle disadvantage is that it is fairly slow when applied to combinatorial problems with a large number of possible permutations, since the algorithm must explore a significant part of the possible solution space during the early phases in order to obtain a result that lies near to the optimal solution

The author has developed a parting line selection technique that uses a simulated annealing phase to refine the solutions which it generates. This is discussed in more detail in Chapter 5.

Chapter 3

Previous Work in the Field

Computer Aided Design techniques are adopting an increasingly important role in the foundry industry, as they have done in many other manufacturing industries. The advantages in terms of reduced lead times and the ability to reuse previous work make them very attractive in the increasingly competitive market place. Research directly related to the problems of foundries has, to date, concentrated mainly on two areas: the modelling of solidification effects to predict shrinkage and stress concentrations during cooling, and the simulation of mould filling effects. Little work has been done on the development of pattern design tools, as distinct from general component design tools.

3.1 Solid Modelling

Solid modelling is an area in which a great deal of research activity is occurring at the moment. It is recognised that a complete description of an object, as opposed to the partial representations provided by wire frame, and even surface models, is a prerequisite if the manufacturing process is to be fully automated. Work is going on to extend both the domain of objects which solid modellers can handle and the level of interactive performance which they provide. Requicha's

classic review papers[49, 48] describe the various approaches used in solid modelling and, although a little old now, still cover all the important methods. A more recent survey of the state of the art, with more emphasis on applications of the technology, can be found in the paper by Woodwark[72]. This discusses various areas where computer methods are employed, including casting design (although much of the work discussed is, in fact, concerned with die casting and injection moulding, rather than sand casting) and N. C. machining. A description of solid modelling techniques can be found in Woodwark[73], which is a good general introduction to the field, although the author's treatment of set theoretic systems is somewhat more detailed than his discussion of boundary modellers. Readers seeking more information about the latter class of system are advised to see Chiyokura[8], which discusses a number of aspects of the implementation of a particular boundary modelling system.

3.2 Feature Recognition

Many of the problems with pattern making are associated with the presence of particular features, such as undercuts, or internal detail, in the casting. If it were possible to detect the presence of such features in the casting and to produce a description of it in terms of the features which it incorporates, this might provide a starting point for some form of expert system for casting pattern creation – it might be possible to transform the problem of finding a pattern set for the casting into one of selecting appropriate solutions to handle each of its constituent features and then combining these solutions. Unfortunately, feature recognition is a hard problem which is current occupying a number of researchers.

Gavankar & Henderson[20], present an algorithm for feature recognition in a boundary modelling system, which is capable of detecting projections and blind holes in a surface. The algorithm locates features by looking for faces containing

multiple edge loops, but is unable to handle through holes. It is also unable to detect projections and depressions which span multiple surfaces, since it works by identifying nested edge loops. The paper also provides a useful survey of the work of a number of other researchers in this area

Woo *et al* [70] have done a lot of work on volume decomposition. This provides a useful technique for detecting pockets and holes, and a slightly more dubious one for identifying bosses and other protrusions on a component. The most interesting aspect of this work, however, is Woo's alternating sum of volumes algorithm, which is capable, for a large class of components, of converting a boundary model of an object into a set theoretic form, by performing union and difference operations alternately on a sequence of convex hulls of parts of the original object. An efficient, general solution to this problem is still a topic for research.

Lee & Fu[19] present an algorithm for feature recognition in a set theoretic model, but they simplify the problem somewhat by not allowing the use of an intersection operator in their modeller, which limits the scope of their proposal somewhat by forcing the models which are to be analysed to be constructed in a manner which is, at times, highly non-intuitive. The algorithm is based upon classification of primitives by principal geometric axes, followed by re-writing of the set theoretic tree to group primitives with shared axes into adjacent nodes in the tree.

Woodwark's[74] survey of work on the problem of feature recognition looks at a number of aspects which make this a difficult problem, and discusses the approaches taken by various researchers.

3.3 Designing with Features

One way to gain the advantages of a feature recognition system without having to devise feature recognition algorithms is to arrange for feature information to

be explicit in the casting model. This can be done by requiring the designer to use a structured input system which builds up an object in terms of the features from which it is constructed: the designer is presented with a list of options for adding, removing and modifying features and builds up the object by applying these operations. Since the feature information is available at this point, it can be retained within the model and used in subsequent operations. Systems like this are often dismissed as too restrictive for real engineering design applications, but a feature based approach does mirror the way in which designers think about the objects that they are creating. An example of this approach can be found in Dixon *et al*[36]. The system described, which they call *Casper*, is written in Common Lisp, and provides a menu driven CAD system, with visual feedback to allow the designer to assess the casting under construction. Castings are built up from a library of so-called *macro-features*, such as U-shaped channels, L-shaped brackets, boxes and slabs, each of which has a number of key dimensions which the designer can vary interactively. These macro features may be built up into an assembly, and *co-features* may be attached to any of them – a co-feature being an attachment or detail which may be associated with a macro-feature, such as a hole, a boss or a rib.

Having constructed a tentative design, the Casper system incorporates a module to evaluate a design's suitability for manufacture. This applies a number of simple design guidelines to a proposed casting design – testing that sectional thicknesses are adequate, that radii are not too sharp and that the overall part size is within acceptable limits. It then performs a simplified analysis of solidification modulii by decomposing the object into its constituent features and computing a solidification modulus for each of them. These modulii are then analysed using a method attributed to Wlodawer[69]: the element with the highest modulus is used as a starting point, and, moving outwards from that point, the ratio of the modulus of each element to that of its neighbours is computed. If a ratio of less

than 1.1:1 is obtained, the element under consideration is flagged as a possible hot-spot. If problems are detected during either the solidification analysis phase or the application of design guidelines, Casper provides the user with an informative message, including a number of suggestions about ways in which the problem under consideration may be minimised or removed. The Casper package makes no attempt to address the problem of parting line selection.

This work was subsequently extended by the same team[35] to produce a system which attempted to handle parametric design problems, where the required design follows a standard pattern, and differs from other, similar designs only in a few key dimensions or components which are selected from a restricted range of possibilities. The system uses an iterative improvement technique, and employs a problem specific scoring function to appraise the tentative design, together with problem specific rules indicating a preferred order in which to make adjustments to the design variables when seeking an improvement. As the authors admit, this type of system is largely confined to design problems which are essentially variations on a common pattern, with possibly a few dimensions or tolerances that vary from component to component, or a limited number of details that need to be devised specially for a particular application.

The gating design system described in Chapter 6 uses a feature based design technique, in as much as the gating features which can be incorporated are selected from a menu of such features, but it is somewhat less ambitious in that it does not attempt to perform an analysis of solidification modulii, and hence it is unable to make any recommendations about ingate or feeder positioning.

3.4 Expert Systems

The field of expert systems is again an area which is the subject of a large amount of ongoing research activity. Traditional expert systems may be classified accord-

ing to the way in which pieces of 'knowledge' are represented within the system. At present, the most widely used method is a *rule-based* approach: information is stored as a series of clauses of the form *if A is true, then B.* A more sophisticated variant of this system can handle so-called *fuzzy logic* by using clauses of the form *if A is true then there is a probability X that B follows.* Systems such as the above have been used in a number of CAD applications,

Neads[41] investigates the automatic assembly of fixtures, using a novel design of fixturing kit, geared towards robotic assembly, and a software package capable of designing a fixture given details of workpiece clamping and location requirements. Whilst this is not directly relevant to the main thrust of this thesis, the process of assembling mould elements into a complete mould is in many ways similar to that of assembling a fixture, and a solution to one problem is likely to provide insights into methods of addressing the other.

Pillinger *et al* [45] describe a system for the automated design of forging dies.

It is based, in part, upon the MODCON system described by Gokler[22], and, as such, has a restricted range of primitives from which the forging may be constructed. The system generates a proposed preform design from a given forging design by applying some standard shape transforming heuristics – the result of this operation is then assessed by an expert system which makes use of a database of previous results. An estimate is generated of how good a preform has been devised, together with a weighting factor which indicates the degree of confidence attached to the estimate – based upon how closely the preform resembles similar preforms present in the database. If the resemblance is insufficiently strong, a finite element analysis is performed to appraise the preform by modelling the deformation processes which occur during forging.

Having obtained an estimate of how good the preform is, the designer can modify the preform, and re-evaluate it, or use it as a basis for the design of preceeding preforms in the forging process.

This is probably the most relevant application of expert system techniques to the pattern making process, but it is really concerned with designing re-usable moulds (forging dies), rather than patterns.

A great deal of work has also been done on the use of expert systems in process planning (a review can be found in Davies *et al* [15])

An area which appears to have been neglected is the incorporation of guidelines on sound casting design principles into a CAD system. This is partly due to the fact that solid modelling systems are not, at present, very widely used in foundries, and, for example, the analysis of solidification moduli requires access to information about mass properties of the component in question, which is not readily available in a less sophisticated modelling system. In addition, the foundry industry represents only a small part of the available market for modelling systems and thus there is less impetus to develop application specific functionality.

More recent developments in the field of artificial intelligence have tended to concentrate on more object-oriented knowledge representations, or on the use of neural nets, but the author is not aware of any work using these representations that is directly relevant to foundry design.

3.5 Modelling of Solidification Effects

This is a topic which has attracted a great deal of attention. In general, either Finite Element (FE) or Finite Difference (FD) methods are applied to predict heat flow. Usually, the FD methods tend to be more useful for producing a quick estimate of the magnitude of shrinkage or stress buildups, whereas the FE methods produce a more accurate result but are time consuming and require significant amounts of computer power and memory for non-trivial problems. Analysis of solidification effects is complicated, when compared with more conventional heat flow analyses, by several factors. These include the release of a large latent heat

of fusion by the melt during solidification (which means that the function relating heat flow and temperature is not continuous), variations in the thermal conductivity of the mould depending upon its exact composition, and the effects of convection within the melt prior to solidification. Reviews of research into the solidification problem can be found in Pehlke *et al*[44].

3.6 Modelling of Mould Filling Effects

Less attention has been devoted to this topic than to the solidification problem, partly due to the greater complexity of the problem. Mould filling is often assumed to be instantaneous, even though it is known that slow or uneven filling caused by a poorly designed gating system is responsible for many casting defects. A more rigourous treatment is given by St John *et al*[28], using a modification of a finite element analysis technique designed for modelling the effects of flood waves. They reported a good correlation between computer predictions and experimental results.

3.7 Gating Design

A number of authors have written about various aspects of gating design, although there is very little on the use of computers in the process.

No work seems to have been done, however, on the integration of gating design rules into CAD packages. A review of the field can be found in Ruddle[52], who surveys a number of papers concerned with experimental and theoretical analyses of the effects of gating shape and position on the quality of the resulting castings. Based on these, he presents some recommendations for gating layouts and runner and ingate profiles, along with some suggestions about areas where further research is still required.

3.8 Tapering

There is very little material on the subject of applying taper to a model automatically, but there has been a considerable amount of research into the related problem of automatically generating smooth blend surfaces between various parts of a model. Notable contributions have been made in this field by Hoffmann & Hopcroft[25], who produce a method based upon potential surfaces, and by Zhang & Bowyer[77], who describe a method, based upon the work of Liming, which allows smooth blends to be constructed between any two surfaces described by polynomial equations. The tapering algorithms described in Chapter 4 use an algorithm similar to Hoffmann's "potential blending" method.

3.9 N C Machining

Since the development of the first CAD systems, people have been investigating the problem of the automatic generation of machining instructions from them. A large number of systems exist which are capable of generating some form of machining program, and the principle area of interest these days is the generation of efficient programs.

Various workers, including Zhang[78], have presented algorithms capable of machining so called *single valued* surfaces: those which can be manufactured on a conventional milling machine with no rotational axes. This is of particular interest in pattern-making, since all pattern elements must be single valued, in order to allow the pattern to be extracted from the mould.

A related area is the problem of tool path planning – ensuring that, as the tool moves around the workpiece, only its cutting surface interferes with the workpiece, and that only at locations where the removal of material is intended. This problem, and the equivalent one of planning the path of a robot arm around a three dimensional space filled with obstacles, have been addressed by many inves-

tigators.

Cameron[7] discusses the problem at length and presents a range of techniques, including volume sweeping and four dimensional modelling, for the planning of robot motion.

Martin[37] discusses the application of envelope theory to the problem of sweeping three dimensional objects through space, and proposes an algorithm conceptually similar to a hidden line removal for addressing the problem.

Davenport[14] considers the mathematics of motion planning and the applicability of the technique of cylindrical algebraic decomposition, concluding that the algorithms presently available are impractical for the analysis of real problems and that more research in this area is called for.

This work also has applications to the problem of automated manufacture and assembly of moulds, although motion planning is far from being a 'solved problem'.

3.10 Cornell Injection Molding Program

Prof. K. K. Wang's team at Cornell have, for a number of years, been looking at the problem of injection moulding of plastics. This is a subtly different problem from metal casting in that the moulds are re-usable, whereas in metal casting they are single shot, with the patterns being the re-usable component. Plastic geometries also tend to be simpler, with undercuts and internal detail being less common. Early work by Wang, Wang[1] and Khullar[64, 65, 34] was carried out using the TIPS-1 solid modeller, principally in the areas of plastic flow during the injection process and optimal placement of gating. They also investigated NC machining algorithms and NC tool path verification, using a depth buffer based technique to provide visual feedback about a proposed machining path. More

[1] W. P. Wang

recent work by the team has concentrated on flow and solidification effects in semi solid metal alloys.

3.11 Forging Preform Design

In the process of forging, metal is shaped by hammering or pressing it against a die in order to make it adopt a new shape. In some ways this resembles casting, although it is closer to the injection moulding process. The problem here is to design a suitable series of formers, or *preforms*, against which the material can be deformed, so as to encourage metal flow in appropriate directions. Work in this area has been carried out by many people:

Bramley *et al* [11, 21] discuss a preform design method based upon a modification of the Upper Bound Elemental Technique (UBET) devised by Kudo, and also present an extension of this method, known as reverse-UBET, which appears to hold promise for automatic preform design, given a description of the target shape, although it is currently restricted to two dimensional problems.

Knight *et al* [4, 22, 63] present a system, again confined to two dimensional problems, which constructs preform profiles from a target cross section, by the application of a number of empirically derived rules.

Yu *et al* [76] describe a system for generating forging die cavities for axisymmetric components, which takes a two dimensional profile, performs some basic machining allowance and draft angle calculations, and generates forging load and billet volume estimates. As with most such packages it is capable of generating N. C. machining instructions for the die cavity. A similar system is also presented by Choi *et al* [9].

Keife[32] compares theoretical and experimental results for a preform design technique which appears to be similar to the reverse-UBET method used by Bramley. The results appear, however, to be somewhat inconclusive.

These systems are all, at present, confined to handling axisymmetric, and hence inherently two dimensional, problems, although Bramley is currently working on extending his work to three dimensions.

3.12 Computer Aided Casting Pattern Design

This appears to be the most neglected area of the casting manufacturing process. To date, the author has located only one significant development in this area: the work carried out by D B Welbourn *et al* at the University of Cambridge. The system which they developed is now available as a commercial product from Delta Cam Systems Ltd under the name DUCT.

3.12.1 DUCT

The DUCT program is designed specifically for the modelling of a particular class of object. It models the object as a series of cross sections perpendicular to a three-dimensional curve. In this it closely resembles the conventional foundry practice of making a skeleton pattern: a series of cross sections of the desired object are manufactured and placed along a centre line. The gaps between these profiles are then filled with clay or resin , which is shaped to form a smooth blend between the profiles. The system is basically a simple surface modeller, which has been developed commercially over a number of years based on feedback from users.

Unfortunately, the DUCT modelling system is not well suited to modelling the types of castings most commonly produced by Listers. Whilst a series of profiles along a curve is a logical way to define a section of pipe-work or a manifold, it is singularly inappropriate for defining, for example, an engine block, which is more readily imagined as the result of joining together a number of blocks and cylinders to produce the desired shape. Objects of this kind are more readily modelled by

a system with some form of set-theoretic input technique, such as the CAEDS modelling system in use at Listers, or the DORA system developed and used at the University of Bath. The development of the DUCT system from a research project into a commercial CAD system can be followed in a number of papers: the earliest reference that the author is aware of is by Morris and Welbourn[39], but this is rather better as a historical perspective. A more useful description of the early DUCT system can be found in Johnson[29] and subsequent enhancements can be followed in [31, 55, 68, 23].

The DUCT system provides some facilities that can assist in the process of casting pattern design. There is a facility which will construct a line joining points of constant surface normal inclination, which can help with the selection of parting lines, but there is no mechanism for automatically joining together a number of such curves to form the boundary of a surface, and no mechanism for making recommendations about good directions in which to separate the pattern elements.

The system does not provide any facilities for evaluating whether a proposed pattern set is well designed, since it has no embedded information about what constitutes a good arrangement and, due to the internal model representation which it uses, it would be difficult to produce volume and mass statistics for the pattern set, and thus to provide information about, for example, the casting moduli for various parts of the mould. This prevents an analysis of the proposed design to identify possible hot spots or other features likely to inhibit succesful, repeatable casting.

3.13 Parting Line Selection

Little work has been done in this area, although Tan *et al*[57] propose an algorithm which can handle simple two part moulds with no undercuts or internal detail.

The approach which they adopt is essentially akin to the 'gift wrapping' technique used in some three-dimensional convex hull algorithms[53], but uses the surface normals of the faces on the object to control how far around the wrapping process proceeds.

In the case of the DUCT program, the surface of the pattern is, modelled as a series of parametric patches 'hung' between the cross sections. The computer model can be used for the generation of cutter paths. It is also possible to compute alternative split lines for the object by specifying the desired direction of separation and then instructing the program to compute the surfaces and lines around it which are perpendicular to the proposed direction of separation. This does, of course, require manual intervention to select a suitable direction of separation and to decide how to split up the model if multiple split lines or cores are required.

Hui and Tan[26] introduce an algorithm capable of constructing patterns for castings which are free from internal detail. They use a technique based on sweeping the casting through space to create the mould cavity, and a ray casting technique to assess possible parting directions. This is one of the more promising efforts in the field, but does not address the problem of cores.

Ravi & Srinivasan[47] present a list of nine criteria which they use to evaluate possible casting pattern designs for injection moulding, but they do not discuss the problem of the weightings which should be assigned to each of these criteria, which is particularly important in this case where, as with most optimisation problems, the solution which scores best in one category often scores poorly in others. This is, however, an important step in the process of developing a practical automated pattern design system.

A general discussion of CAD in the foundry can be found in Welbourn [68] and in Roberts [50]. Of less direct relevance, but also of interest, is the work by Dardiry *et al* on the use of group technology to reduce casting scrap rates[12]. A review of various foundry CADCAM projects, mainly in the United States, can

be found in Clegg[10].

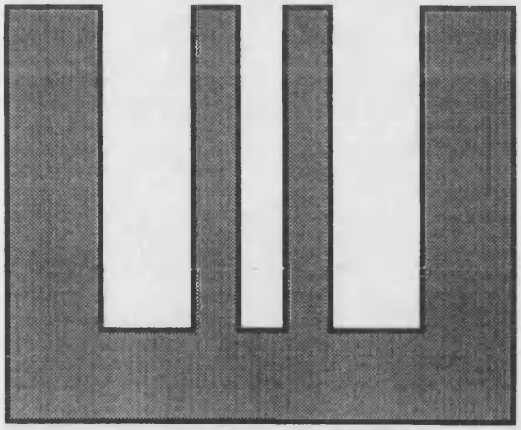
Chapter 4

Tapering of Castings

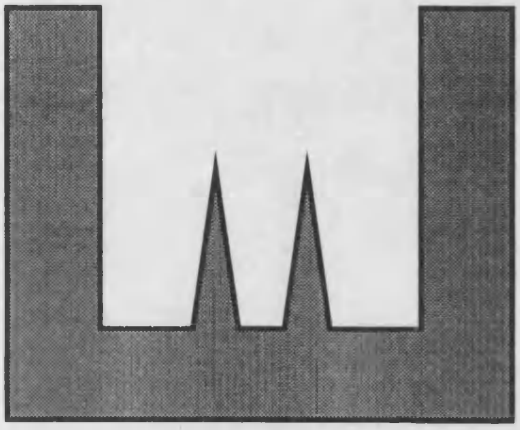
4.1 Desired Behaviour

Tapering is essentially a local modification to a casting pattern to simplify the manufacturing process. It is desirable that the effects of the change to the object be restricted to as small a volume as possible. In particular, taper should only be applied to surfaces which are close to vertical.

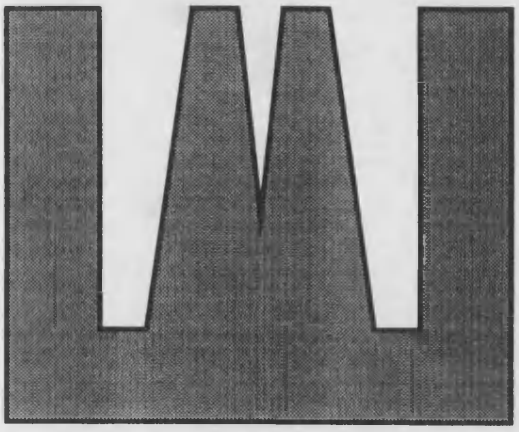
Ideally, the tapering process should not alter the geometry of the part significantly. Unfortunately this last requirement can be difficult or impossible to achieve sometimes, particularly where deep, narrow slots are present in a casting (for example, cooling fins on an air cooled device tend to be of this form). This problem is illustrated in Fig 4-1. If taper is applied outwards from the top, then the gap at the bottom between fins will be closed up. If it is applied inwards from the bottom, then the top of the fins becomes very thin and difficult to cast reliably. If one attempts to get around this by applying less taper, then it may be difficult to produce moulds reliably in less resilient, but cheaper, moulding media, such as green sand. To avoid these problems, a more expensive casting process, such as shell moulding or investment casting, may be required. Both of these casting methods allow the use of smaller degrees of tapering on the pattern set,



Original Object



Tapering to preserve cavity truncates fins



Tapering to preserve fins fills in cavity

Figure 4-1: Problems tapering deep cavities

in the former case because the moulding medium uses a resin binder, and can therefore withstand rougher treatment, and in the latter because the pattern is not removed from the mould; instead it is composed of a material (often wax) which is either melted out of the mould prior to pouring, or is vapourised by the heat of the incoming molten metal (which does, of course, mean that the pattern is not re-usable).

4.2 Existing approaches

Tapering is often controlled by the pattern maker; the engineering drawings will not stipulate any taper and it will be up to them to choose where to apply taper and how much to impose. One obvious disadvantage with this is that two patterns for the same part, produced by different pattern makers from the same drawing, may differ in the degree and position in which they choose to apply taper, possibly to the point where they have different requirements for subsequent machining. This is undesirable from the point of view of an integrated CIM system, since subsequent processing operations are required to take into account variations in the position and magnitude of flash on the casting. In particular, the fettling requirements of components produced from different patterns may be very different, which makes it difficult to automate the process.

Another approach which has been adopted is the use of tapered milling cutters. Essentially, the pattern is machined from solid, normally in accordance with an NC part program which is generated from a model of the pattern with no allowances for taper on it at all. A cutter which tapers slightly towards the tip is employed, but the cutter description which is supplied to the tool path generation software is for a straight-sided cutter. As a result, faces which are vertical in the component model will actually be cut with a small amount of taper on them. This technique can be very effective but it means that there is no accurate computer model of

the pattern.

This would inhibit, for example, the automatic generation of a program for an NC coordinate measuring machine for automated inspection of the casting[62].

4.3 Approaches Investigated

The author has developed a method of tapering an arbitrary object represented by a set theoretic model composed of polynomial half-spaces. The algorithm is equally applicable to a boundary modeller, but no implementation has been produced for such systems since the author does not have access to one for testing purposes.

The algorithm works by taking the equation of the primitives which make up the object, which in general are of the form:

$$F(x, y, z) \leq 0$$

and replacing them with equations of the form:

$$F(x, y, z) + G(x, y, z) \leq 0$$

If we are applying taper to a flat surface, G is a linear function of x, y and z, ie

$$G(x, y, z) = rx + sy + tz + u$$

and the equation of the primitive will also be a linear function, having the general form:

$$ax + by + cz + d \leq 0$$

In this case the surface normal of the plane is the vector:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

and the plane is a distance d from the origin at the closest point[1].

If we apply the algorithm to this primitive it will be replaced by one with the following equation:

$$(a+r)x + (b+s)y + (c+t)z + (d+u) \leq 0$$

By analogy with the above case it can be seen that this corresponds to a planar half space whose surface normal is the vector:

$$\begin{pmatrix} \sqrt{\frac{(a+r)^2}{(a+r)^2+(b+s)^2+(c+t)^2}} \\ \sqrt{\frac{(b+s)^2}{(b+r)^2+(b+s)^2+(c+t)^2}} \\ \sqrt{\frac{(c+t)^2}{(a+r)^2+(b+s)^2+(c+t)^2}} \end{pmatrix}$$

and which is at a distance:

$$\sqrt{\frac{(d+u)^2}{(a+r)^2+(b+s)^2+(c+t)^2}}$$

from the origin at the closest point. This shows that, by varying r, s and t appropriately, one can adjust the inclination of the plane in any direction and by varying u one can control its distance from the origin. Thus by suitable selection of the values r, s, t and u one can taper the plane in any direction whilst leaving any chosen point on it in its original position.

[1] if the vector a, b, c is normalised – ie if $a^2 + b^2 + c^2 = 1$

The selection of a suitable invariant point when tapering the surface has a significant effect upon the appearance of the tapered object. The simplest strategy from an implementational point of view is to pick a particular plane perpendicular to the proposed separation direction and to alter all the planes in the model so that the line of intersection with the chosen plane remains invariant (see Fig 4-2).

This method has the advantage that it is not necessary to consider the position of the faces on the model, but this is also one of its drawbacks, as can be seen from Fig 4-3.

Because a single plane is invariant, points further from that plane will be displaced horizontally, giving a 'shoulder-widening' effect as shown. This suggests that some form of localisation of the tapering effect would be advantageous.

4.4 Localisation of the Tapering Effect

In order to attempt to localise the tapering effect, and thus avoid the 'shoulder widening' problem described earlier, it is necessary to establish the edges which bound the vertical, or near-vertical, faces in the model. In a set theoretic modeller this is a fairly expensive operation, since it is necessary to use most of the steps required to translate from a set-theoretic to a boundary model in order to obtain the necessary edge data. This suggests that a boundary model might be a better starting point for this type of transformation. The author is inclined to disagree with this view. Tapering is essentially a local modification to the structure of a solid model and it is when performing such modifications to a boundary model that there is the greatest risk of invalidating it (ie of generating a model of an object which cannot exist).

One should,however, point out that the amount of taper applied to castings is small – normally only a few degrees. As a result, a lot of these problems do not

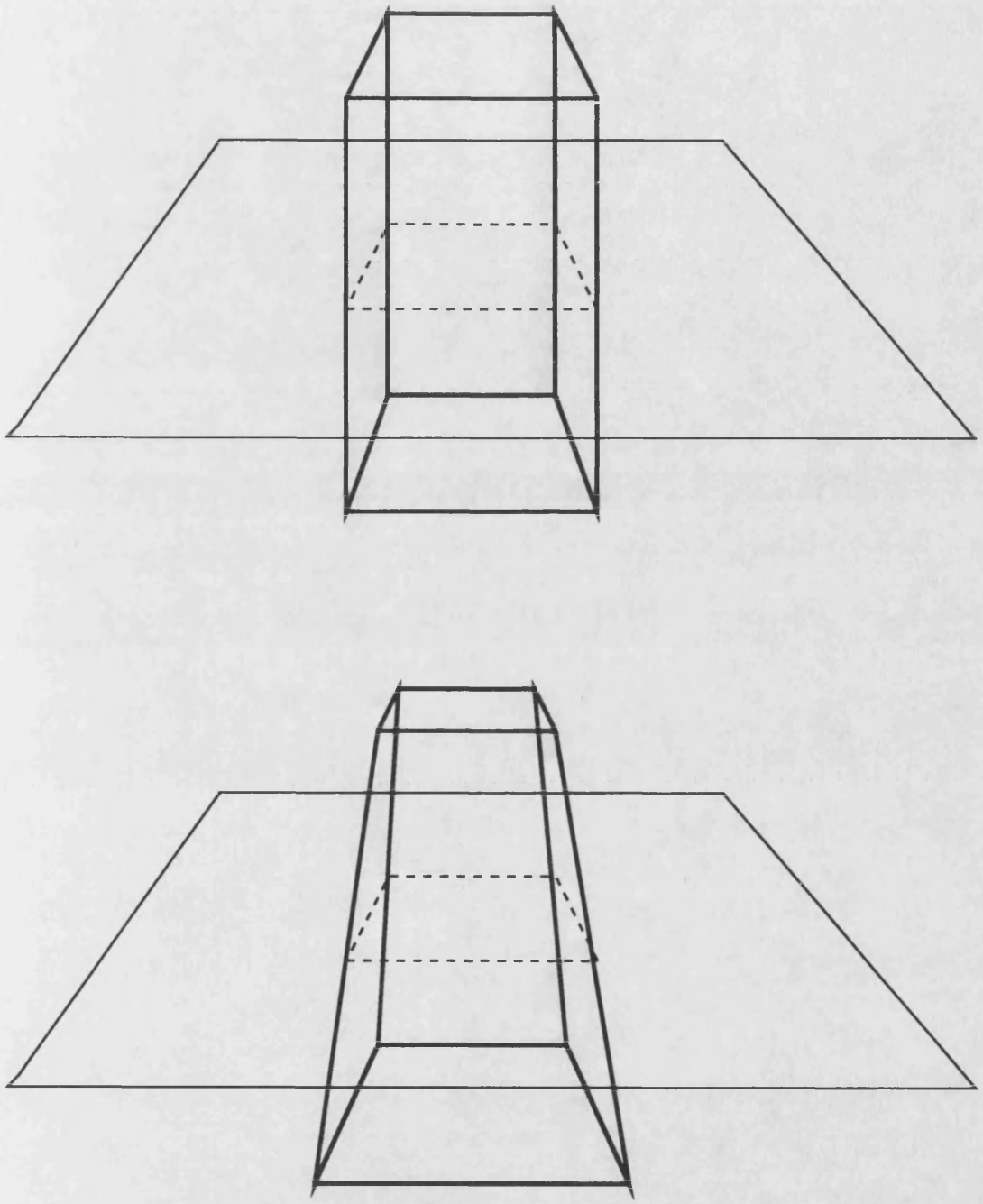
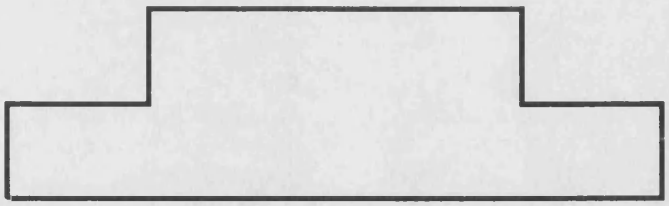


Figure 4-2: Tapering with a single invariant plane



a) Original Object Profile

b) Ideal Tapered Profile

c) Actual Tapered Profile - Note the ''Shoulder Widening'' effect

Figure 4-3: The *shoulder-widening* problem

occur in practice – in a good casting design the designer will be aware of the need for taper and will avoid narrow slots unless the design has a functional requirement for them. The most common example of such a functional requirement is probably cooling fins, especially on air cooled apparatus. These are most commonly handled by a process requiring lower levels of taper on the pattern – such as shell moulding.

The author has implemented an algorithm for tapering set theoretic models. The implementation will apply taper to the faces of models constructed from planar half spaces. It uses the same model format as the DORA modelling system used at Bath. It uses a modification of the MEG (Model Edge Generator) program described elsewhere [71]. The algorithm proceeds as follows: for each planar half space within the model which has a surface normal which is close to perpendicular to the proposed separation direction, a list is constructed of all the vertices that lie on the half space. This point set is then used when selecting a line on the half space which will remain unaffected by the tapering process. The program offers the designer three strategies for tapering a given face: one can leave the top vertex, the bottom vertex or the centroid of all the vertices on the half space in its original position, rotating the rest of the half space about it. Clearly the provision of a choice of three tapering strategies is somewhat arbitrary, but it allows the user to cover the range of useful tapering options in a reasonably small number of trials; since it is easy to construct test cases which are handled badly by any one of the above strategies, the author would suggest that the best procedure would be to try all three possibilities on a casting and to adopt the one which appears most satisfactory.

The observant reader will have noticed, in the above discussion that there was no mention of models used by the DODO system, ie models incorporating curved surfaces. This is because the algorithm as implemented is restricted to handling planar faces – essentially because of the need to determine the boundaries of the faces in order to localise the tapering operation. In principle the algorithm can

be extended to handle curved surfaces by computing a few 'representative' points on the surface.

When dealing with curved surfaces one possible approach to the tapering problem is to ignore it: curved surfaces will not have large areas parallel to the separation direction and so will not need tapering. This is unsatisfactory in that it does not address the problem of, for example, a relatively common engineering feature – the vertical cylinder. Vertical cylinders are, however, fairly easy to detect, and to taper by converting them to conic sections. Thus the most common problem case could be handled with the addition of a small amount of special case code.

4.4.1 Tapering Higher Order Surfaces

Some complications arise with higher order surfaces. When tapering a planar half space, it is sufficient to add to the surface equation a term which varies linearly with distance along the proposed separation vector. This does not work so well for curved surfaces. If we consider the case of tapering a cylindrical half space along the z axis, and, for simplicity we consider the particular case where the axis of the cylinder is alligned with the z axis, we have the governing equation:

$$ax^2 + by^2 + r^2 \leq 0$$

where r is the radius of the cylinder. Ideally we want r to vary linearly as one moves along the z axis, *ie*

$$r' = \alpha + \beta z$$

For some constant α and β.

This gives a modified equation of the form:

$$ax^2 + by^2 + r^2 + (-r^2 + \alpha^2 + 2\beta z + \beta^2 z^2) \leq 0$$

Which is quadratic in z. In general, it is possible to compute a tapering function of this form for any polynomial surface, with the order of the function being no higher than the order of the surface being tapered.

A few examples of the algorithms in action can be seen in Figs. 4-5, 4-6 and 4-7.

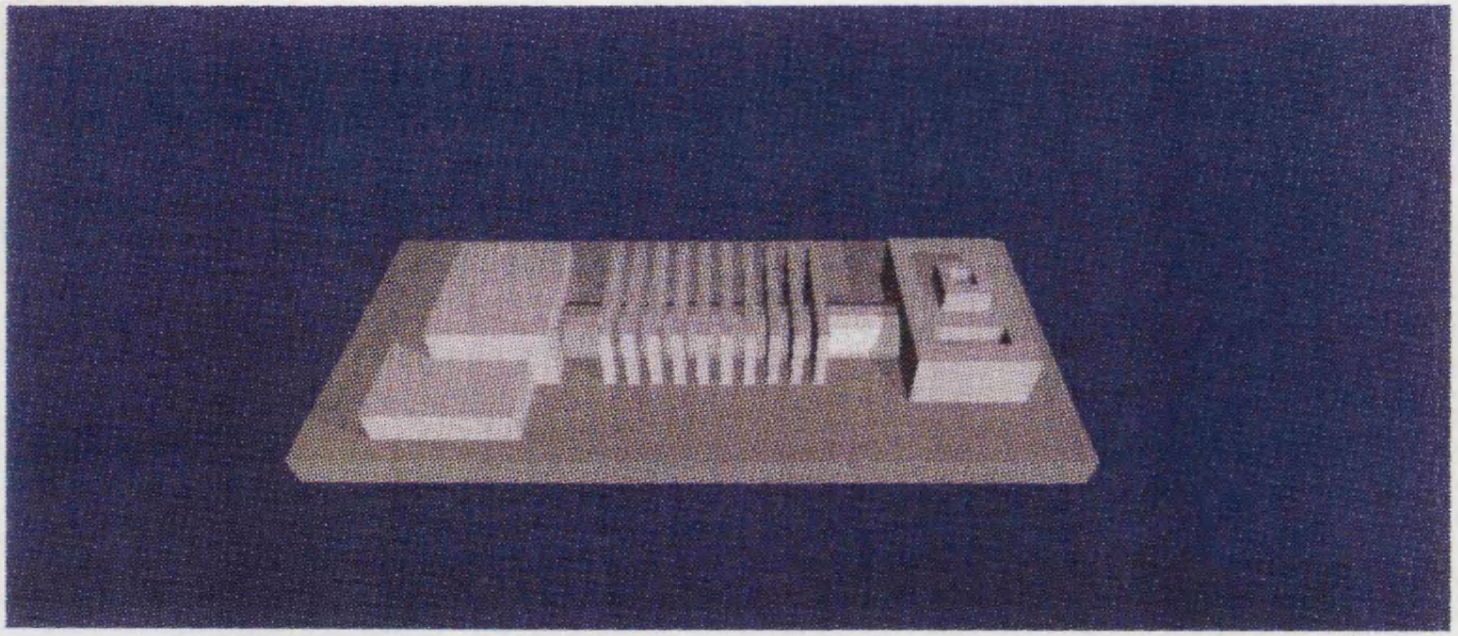


Figure 4-4 Tapering test case (above)

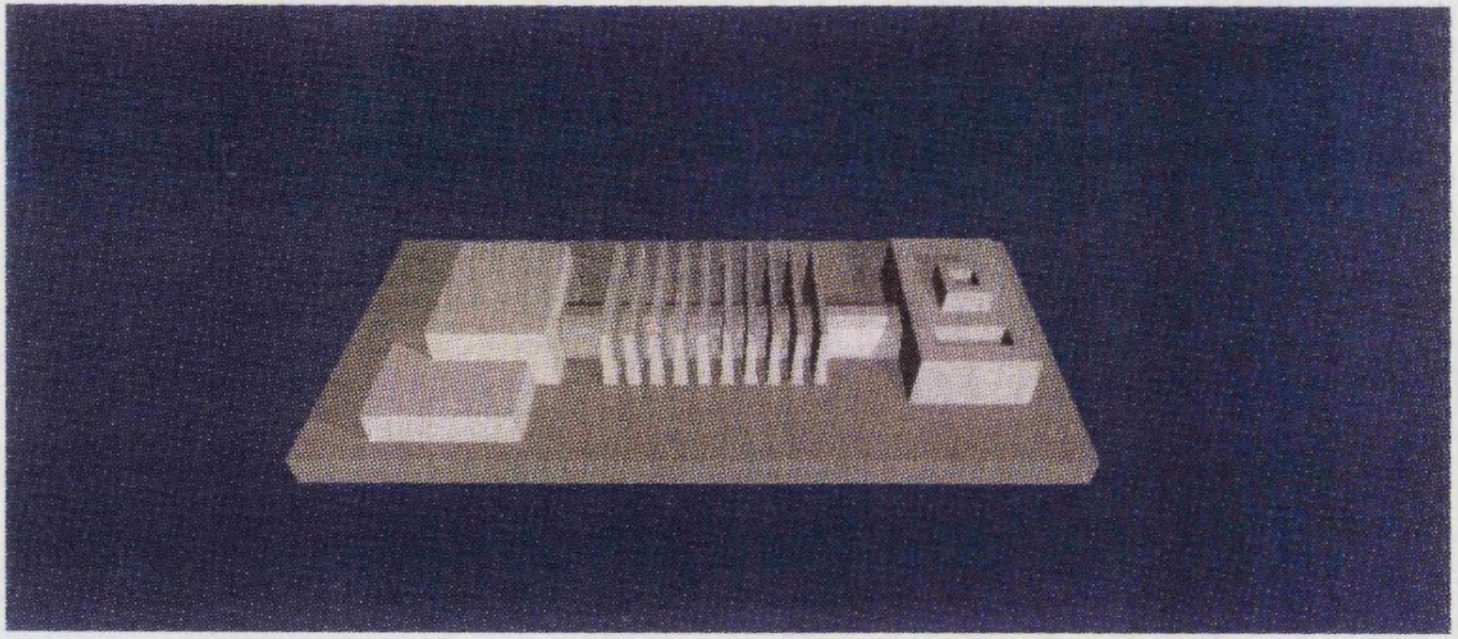


Figure 4-5 Mid-point based taper (below)

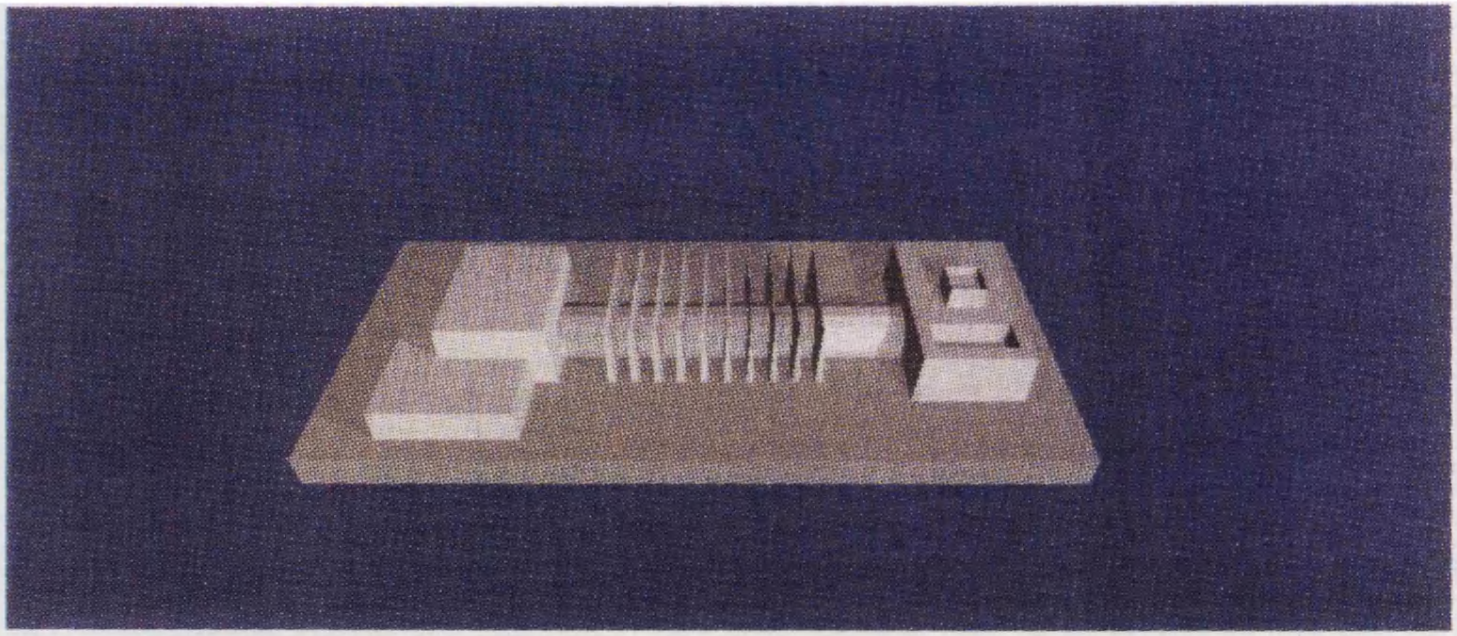


Figure 4-6 Bottom based taper (above)

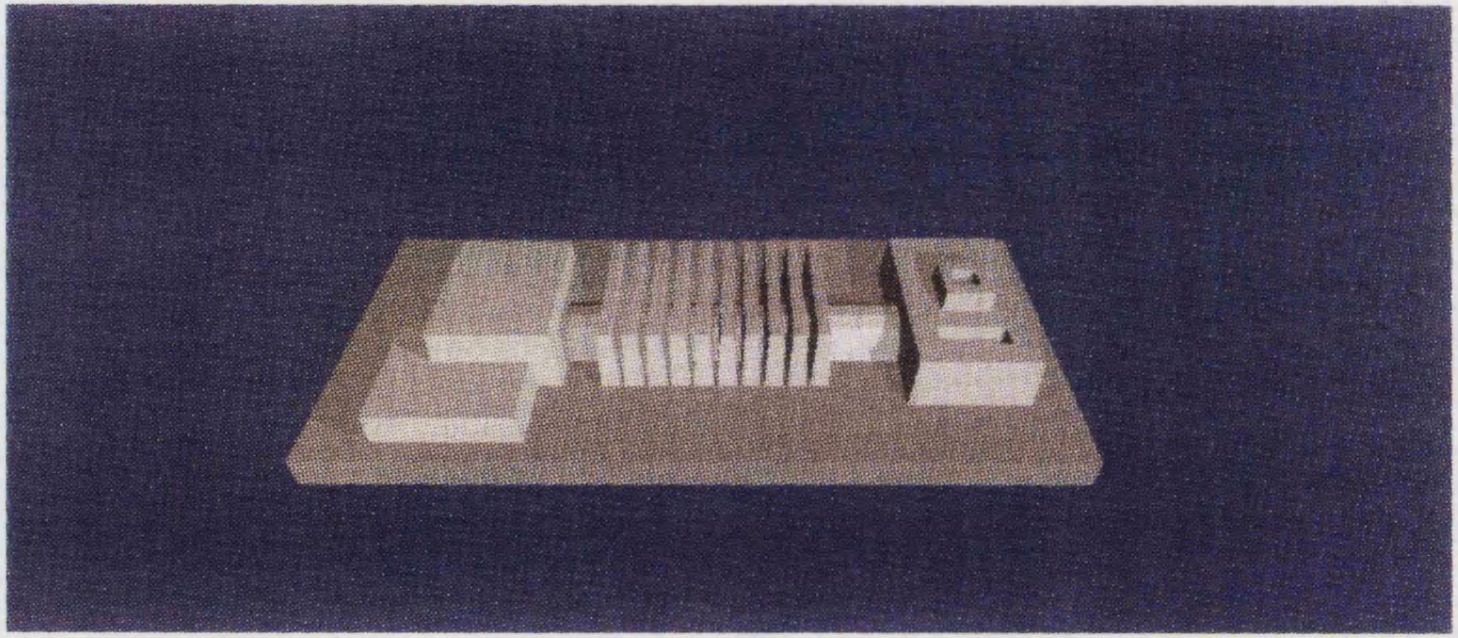


Figure 4-7 Top based taper (below)

Chapter 5

Determination of Parting Lines

In order to be able to construct the individual pattern elements that make up the pattern set for a given casting, it is necessary to decompose the casting itself into a number of regions. Each region must possess the property that there is at least one direction vector $\bar{v}$ such that all normals to those surfaces of the casting that lie within the region are at a relative angle of no more than 90° to $\bar{v}$. Any such region can be considered to be a suitable candidate for turning into a pattern element, since it is possible to pack sand around it and then to free it from the mould element thus formed by withdrawing it in the direction $-\bar{v}$ without disturbing the sand in the mould (Fig. 5-1). It is, of course, possible that, for a given region, there will be a range of direction vectors satisfying this requirement. It is also usually the case that there are several alternative schemes for grouping the object into regions which would give rise to different pattern arrangements. The vector $-\bar{v}$ is also the direction in which taper must be applied to the model in order to assist with the separation process. It is referred to henceforth as the *separation vector.* The boundaries between regions defined in this way are known as *parting lines.*

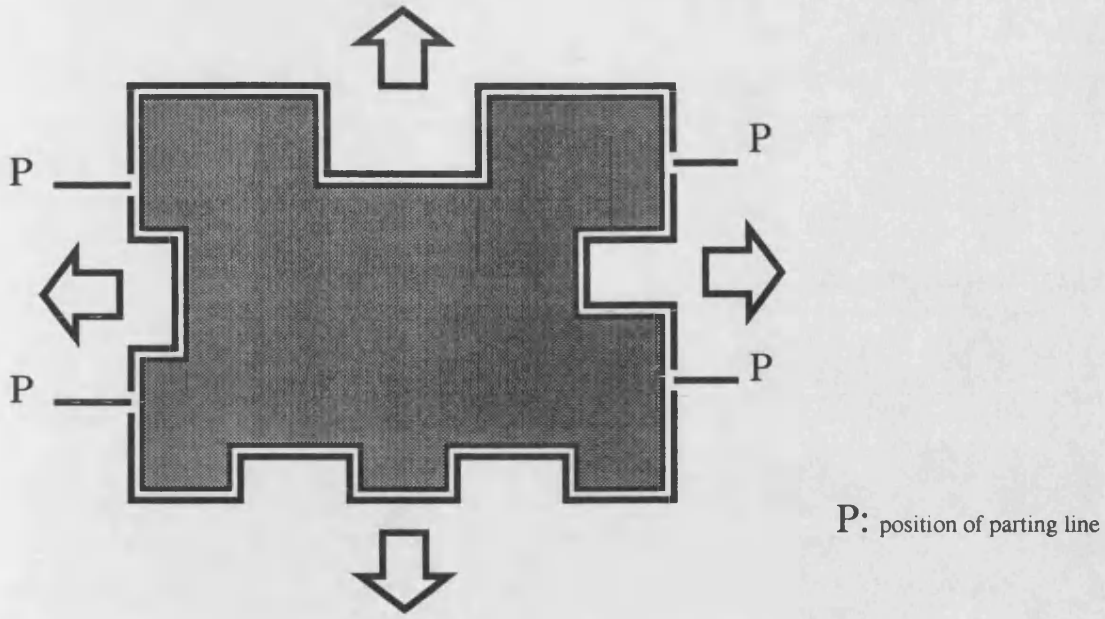


Figure 5-1: Parting lines on a casting

5.1 Desired Behaviour

When selecting a parting line a number of goals must be bourne in mind. Firstly it is desirable to have as few components in the pattern set, and hence in the resulting mould, as is possible. This reduces the cost of manufacturing the pattern set, and also reduces the cost of each mould, by reducing the number of components which need to be assembled.

It is also useful to ensure that the mould does not contain any small pieces; these are, in general, harder to handle and easier to omit inadvertently, as well as being more prone to breaking up or being displaced from their intended position in the mould as molten metal is poured in.

5.2 Existing Approaches

Current practise in the foundry industry is generally to leave the process of pattern split line determination to skilled pattern makers, who make decisions based

upon their personal experience. The success of this method is, of course, completely dependent on the level of expertise of the pattern maker. As stated earlier, problems can arise due to variations between one pattern maker and another and to human error, particularly given that three-dimensional assembly problems are hard to visualise. This difficulty is further compounded by the requirement for a double inversion of the geometry of the casting in the process of translating from casting to mould and thence to pattern.

Very little is available at this time by way of casting pattern design software, but some systems, most notably the DUCT package discussed in Chapter 3, do make an effort to provide the pattern maker with some tools to assist with the tasks peculiar to this operation.

5.3 Algorithms Implemented

The design of a pattern is complicated by the fact that, because it is necessary to withdraw the pattern element fully from the mould, the decision about which direction is suitable for withdrawing pattern parts is influenced by non-local effects – it is not sufficient to be able to remove the pattern from the cavity it has created: a path must exist which allows it to be withdrawn fully from the mould without interfering with the mould at any point. The calculation of such an extraction path is a hard problem, equivalent to the robot motion planning problem, which has received attention from numerous researchers, including Cameron[7], Martin[37] and Davenport[14]. For the purpose of this project, the problem has been simplified by assuming that extraction is impossible unless it can be performed by moving the pattern in one direction only until it is fully outside from the region enclosing the mould. This restriction greatly simplifies the testing of a possible separation direction, but it means that a number of potential solutions may be rejected unnecessarily; in particular, it effectively prevents the system from ever

using a loose piece to create an internal feature in a mould cavity – it will, instead, always choose to use a core in this situation. This is not necessarily a bad thing, however: it seems likely to the author that a design based upon the use of loose pieces would be less suited to an automated assembly process than one which produced the same features by the use of cores; this might tend to inhibit the subsequent development of an integrated automated casting production system.

5.3.1 General Approach

The splitting algorithm works as follows.

It starts with set theoretic model of the desired casting and a cuboidal volume of space, alligned with the coordinate axes, which is known to enclose the object completely. This volume is divided into a number of small boxes such that each box thus obtained passes a basic separability test. Each box is then subjected to a more rigourous test in order to obtain a list of the directions in which it could be separated. The boxes are then grouped together in clusters of boxes with a shared separation direction. For each of these clusters of boxes, a solid model is generated – of the parts of the casting which the current cluster encloses. These models are then written out in a format suitable for subsequent processing by the other modelling tools developed at Bath. The following sections discuss this process in more detail.

The output from this algorithm consists of a number of solid models which define the separate elements of a mould for the casting. By differencing these models from suitably sized and positioned blocks of material, it is possible to construct a model of the pattern to produce that part of the mould.

Division Phase

The volume enclosing the object to be cast is divided into sub-boxes by a recursive algorithm. This algorithm works by selecting a suitable plane, parallel to one

of the axes, which passes through the box under consideration. This plane is used to divide the box into two new sub-boxes, one on each side of the plane. The original object description is then pruned to each of these sub-boxes (in the manner described below), to produce two new object descriptions, one for each of the sub-boxes. Each of these will, ideally, be a simplified version of the original model which is equivalent to it within the sub-box under consideration. The sub-boxes are then tested, in turn, to establish whether they are sufficiently simple to allow subsequent stages of the process to be performed. If this is not the case, then the sub-box is divided again recursively until a suitably simple set of sub-boxes has been produced, otherwise the current sub-box is adopted. The result of this process is a binary tree in which the non-leaf nodes are planes about which the original box has been divided and the leaves are simplified models of the original object, each of which is equivalent to the original object, but only within a limited region of space.

The division process requires a testing phase to evaluate the sub-models and decide if they are sufficiently simple; otherwise it would continue indefinitely (or, at least, until the available memory was exhausted). In this case the division process is terminated when the current box is considered 'separable' or when the box under consideration is smaller then a limiting value, derived from the volume of space occupied by the whole object.

Two basic division strategies are available. The first uses even binary division: the box under consideration is recursively divided into two equal parts along the longest edge. This gives similar results to oct-tree division, but tends to behave better if a poor division criterion is employed, causing the division process to proceed further than necessary (simply because binary division creates one new box at each stage, whereas oct-tree division produces seven). The price of this improvement is a slight increase in the time taken to traverse the divided model in order to reach a given sub-box.

In an attempt to reduce the number of boxes in the set of parts requiring processing, a more complex division strategy was introduced.

The primitives within a sub box are considered in turn, and the normals to the surfaces of the primitives are compared to the coordinate axes. If a primitive has surfaces on it that are, at some point, normal to one of the axes, then a counter associated with that axis is incremented. When the primitives within the sub-box have all been considered, the box is split along the axis whose associated counter has the largest value. If two or three axes have the same associated score, then the split is introduced along whichever of these axes it is that corresponds to the largest sub-box side length. The idea behind this strategy is to introduce division planes which lie between primitives, rather than intersecting them.

This works well for objects composed of planar facets, since the division plane will tend to be parallel to the majority of half-spaces in a box, and so will tend to do a better job of separating half-spaces into one box or the other, rather than having them straddle both. Its value when applied to curved primitives is more questionable, although it should still work well for objects which are predominantly composed of flat surfaces and it is, in any case, no worse than simple binary division for most objects. It should, also tend to perform better as the boxes become smaller, since the surface curvature will decrease as successively smaller and smaller regions of the curve are considered.

This technique could be enhanced further by controlling the position, as well as the orientation, of the next division, based upon the distribution of half spaces within the box, possibly by using a mechanism similar to the *Zone Division* algorithm employed by Woodwark[71]

This algorithm is based upon the concept that, for the purposes of division, two points along one of the coordinate axes are equivalent if they both have the same inside/outside relationships with all half spaces within the sub-box currently under consideration. Firstly, one determines, for each half space, the x, y, and

z values at which a point can be guaranteed to be inside the half space, and similarly the values at which points are definitely outside the half space (effectively computing the bounding box which encloses the half space within the sub-box under consideration). Given this information, one can combine these 'bounding boxes', by working upwards through the set-theoretic expression describing the model. The result of this operation is a list, for each of the coordinate axes, of coordinate values at which the effects of introducing a division plane change, and, for each region between these coordinates, an estimate of the number of half spaces to each side of the division plane (for a model composed of planar half spaces, these estimates will be exact). By comparing the estimates thus obtained, one can select the most suitable point at which to introduce a division plane (Woodwark's implementation chooses the division plane which minimises the sum of $N_{hs}^2 A_s/2$ for the resulting two sub-boxes, where N_{hs} is the number of half spaces in the volume and A_s is the surface area of the volume).

Separability Test

The separability test works as follows: the primitive half spaces within the box are considered in turn and an estimate is produced for the range of surface normal values possessed by points within the box on the surface of each half space. For a planar half space this is simple and exact, since at all points on the boundary of the half space the surface normal will be the same. In the case of a non-planar half-space this is not true and a more complex technique must be used to compute upper and lower bounds on the range of x, y and z components which the surface normal vector will possess.

One possible approach to this problem is the use of interval arithmetic, but, for complex surfaces the bounds on the surface normal vector that this method gives are excessively wide. Other methods do, however, exist. An alternative approach is to use the Bernstein Transform. This technique is discussed in Davenport[13]

and also in Milne[38], where the steps required to extend it to N dimensions are spelled out. In outline, one re-writes the polynomial describing the surface normal in an alternative form. This form posesses the special property that, if one takes the coefficients of the terms in this representation, and selects the smallest and largest values, one obtains bounds upon the range of values which the polynomial can take which are tighter than the bounds given by the use of interval arithmetic (the proof is basically similar to the proof of the convexity property for the control points on a B-Spline curve: the expression is a weighted sum of expressions where the weights sum to unity for all values of the free variables)

The algorithm only considers separation of the casting along the co-ordinate axes, in the $+x$, $+y$, $+z$, $-x$, $-y$ and $-z$ directions. This is clearly a restriction but is less severe than might at first be thought, since engineering components are normally designed in a manner which encourages alignment of features with the co-ordinate axes. It is also trivial to preprocess the model by asking the user to nominate a 'preferred principal separation direction' and then to change the orientation of the model so that this preferred direction is alligned with the coordinate system.

For each of these possible separation directions a test is carried out against all the surface normal ranges obtained previously. If none of the half spaces in the box has surface normal components pointing in the opposite direction then separation direction under consideration is added to a list of acceptable directions for the current box. If, however, there are surface normals pointing in an opposing direction then separation in this direction is not possible. An example may help to clarify this.

Suppose we have divided and pruned (See page 40) an object and obtained a sub-box $((x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max}))$. Within this sub-box the object might be represented by a set theoretic model with three primitives, A, B, and C, all of which are planar half spaces. We compute their surface normals and get

three vectors, for convenience, (0, 1, 1), (0, 1, 0) and (-1, -1, 0). If we consider separating the object in the $+x$ direction, we proceed by testing the x components of these vectors. The first two present no problem, but the third has a negative x component, which means that separation in the $+x$ direction is not possible for this sub-box because of primitive C. Repeating this process for the other five possible separation directions that we consider we conclude that this sub-box could separate in the $+z$, and $-x$ directions.

A box is considered separable if it can be separated in at least one direction along each of the co-ordinate axes ie ($+x$ or $-x$) and ($+y$ or $-y$) and ($+z$ or $-z$). This is an over-simplification, of course – a box is only separable if it can be fully withdrawn in the proposed direction, so a box which is classified as separable at this stage may, in fact, be unable to travel in the proposed direction, due to the presence of a feature in some other box along its intended path which would obstruct motion in that direction. It is because of this factor that three perpendicular separation directions are required at this stage: subsequent stages of the algorithm may rule out some of the possibilities and it is necessary to have alternatives in order to improve the chances of constructing a pattern successfully. If, however, the size of the box under consideration becomes sufficiently small compared with the original object space, this requirement for three different separation directions is relaxed, and the division process will be stopped if a single separation direction can be found. This reasoning here is that, if this box is small, then repeated divisions and pruning operations have already been carried out on it without producing anything that is separable in three directions. In this case, although it is perfectly possible that one more division will produce two separable regions, it is quite likely that a feature has been encountered which has a very restricted range of possible separation directions. Rather than complicate subsequent processing by increasing the number of sub-boxes in the model further, a reduced acceptance criterion is adopted.

Classification Phase

At the end of this process, the algorithm produces a list of boxes, each of which has an associated list of the separation directions which are compatible with the half spaces that intersect that box. It is now necessary to determine, for each of these boxes, whether it is in fact possible to separate it in the proposed direction – it may be the case that a box near the middle of the object can be separated in the $+x$ direction, but that there is another box, next to it in the $+x$ direction, which will only go in the $-x$ direction. This latter box would prevent the former box from actually separating in the $+x$ direction (assuming, as stated earlier, that we only consider straight paths for the removal of pattern elements). In order to confirm that a proposed separation direction is actually useable it is necessary to construct a list of the sub-boxes that lie wholely or partly within the volume swept out by the box currently under consideration as it travels in the proposed separation direction. One then tests any other box which appears on this list to confirm that its possible separation directions include the one currently being considered. These tests can be performed quite quickly by exploiting the tree structure of the divided model produced by the previous stage and the fact that the division planes, and hence all the sub-boxes, are aligned with the co-ordinate axes – thus a simple comparison at each node in the tree will indicate whether one or both branches need to be examined for boxes intersecting the separation box.

The result of this phase of the process is a list of boxes, each of which has an associated list (possibly empty) of the directions in which it can be separated.

At this point it is necessary to group together those boxes which are adjacent to one another and which share a common possible separation direction into clusters. These will form the basis of the individual pattern elements.

As a first stage, the list of boxes is sorted, based upon the number of possible separation directions each box has. This is intended to ensure that the boxes

with the fewest possible separation directions are processed first. This is because there will normally be several alternative separation directions for any box, and the choice of the best direction is influenced by the directions which have been chosen for other boxes, since one objective of the process is to minimise the number of different directions in which the object is separated. By processing the elements which have the most restricted separation directions first, one can identify particular directions in which some part of the pattern must move, and then use this information in selecting the preferred separation direction for less heavily constrained boxes.

The boxes are grouped together into *clusters* in the following manner: the system maintains seven lists of clusters, where a cluster is a list of boxes which are connected to one another by shared or overlapping faces. These seven lists are of the clusters that will separate in the $+x$, $+y$, $+z$, $-x$, $-y$ and $-z$ directions, and of the clusters for which no separation direction can be found (usually because they form part of some internal detail). The algorithm considers each box on the (sorted) list of boxes produced by the previous phases of the process and tests the effects of adding it to each of the first six cluster lists (unless it has no possible separation directions, in which case it is automatically added to the seventh list). Adding a box to a cluster list is a slightly involved process, since the box may be share all or part of one of its faces with boxes already on the cluster list. There are essentially three possibilities; the new box may not touch any of the boxes on the cluster list – in this case the box forms the start of a new cluster; it may touch boxes in only one cluster – in this case the box should be added to that cluster. The third possibility is that the box touches boxes in more than one cluster on the cluster list – in this case all of the clusters which it touches should be merged together into one cluster, and the new box should then be added to it.

The algorithm uses a cost function (See Section 5.5) based upon the number of clusters and the volume of the object which they make up to assess each alter-

native. The cost of a cluster list is computed both before and after adding the box under consideration and the difference between the two is used to measure the net cost of adding a box to the current cluster list. The box is then added to whichever of the seven cluster lists has the lowest associated cost. This is of course a local, rather than a global, optimisation strategy and so may not produce a truly optimal grouping of boxes into clusters, but the pre-sorting of the list of boxes compensates for its more obvious deficiencies.

At the end of this process the algorithm produces seven lists of clusters; the first six lists are of regions of the model which are to be separated in the $+x$, $+y$, $+z$, $-x$, $-y$ and $-z$ directions, and the seventh is of pieces which could not be separated, due to interference between the piece in question and some other sub-box in the divided model during the separation direction testing phase.

Model Generation

Processing of the first six lists is completed by writing out a solid model for each cluster on each cluster list. This model is formed by constructing a set theoretic model of the boxes in the cluster all unioned together. This defines the volume of space which this cluster occupies. The model thus obtained is then intersected with the original model of the object to be cast – effectively 'clipping' the original model against the proposed parting surface.

Since the clusters are basically a list of adjacent cubes, a model constructed from them will consist of a large number of coincident half spaces. These tend to stretch the capabilities of algorithms used to process the model, so at this stage the opportunity is taken to eliminate some of them by using a pruning operation (which eliminates from the model those boxes which contain only air, at a saving of six half spaces per box eliminated).

Handling Cores

This process will, in general, leave a list of clusters on the seventh list for which no separation direction could be found owing to interference problems. These are the parts of the casting for which cores will be required. They are handled as follows.

Each cluster on the cluster list is considered in isolation. For each box in the cluster there exists a list of the possible separation directions for that box which are compatible with the primitives of the original object inside it. The interference testing process used earlier is now repeated, but this time only collisions with other boxes within this cluster are counted. The steps described above for grouping the boxes together into clusters are then performed, again considering only the boxes that are in the current cluster. This will produce a number of models for the parts of the casting that are to be handled with cores and may produce a list of clusters which are still not handleable. These clusters can be processed by repeatedly applying the above process for clearing the seventh list to the clusters which remain on this list after each phase.

In this way a set of models of parts of the original object is produced. Each model corresponds to a single pattern element – its boundaries correspond to the position on the original casting where a parting line would appear.

It should be noted that this process does not guarantee that all pieces of the original model which lie within a single cluster are connected. This is because it is possible for the pruning code to fail to identify that the sub model for a given box is equivalent to a null model (this is unusual, but quite possible, since it performs conservative, rather than exact, tests). Any such box should not be used during the cluster assembly process to link together separate clusters, since the box does not correspond to a piece of metal spanning the two clusters. It is difficult to detect when this problem has arisen, and the current implementation does not attempt to do so. As a result the algorithm occasionally produces pattern elements

containing multiple, independent cavities, which are used to produce more than one mould element, which is unsatisfactory, but not incorrect.

5.4 Experimental Results

The splitting algorithm was applied to a number of test cases (See the colour plates at the end of this document). The results of this operation can be summarised by saying that the code produces good results for simple test cases, even when the test model incorporates features requiring the use of cores, but that it tends to behave less well on more complicated objects, normally producing solutions that are more complicated than a skilled pattern maker would devise. The tests are considered in more detail below:

5.4.1 Test 1

(see Fig. 5-2 & Fig. 5-3)

This is a simple, solid cube of material. As one would expect, the algorithm classifies this into two parts – one to separate in the $+x$ direction, one in the $-x$ direction. In the diagram, elements in shades of blue are real surfaces on the original casting, whereas those in shades of green are the boundaries between regions which are classified as separating in different directions.

5.4.2 Test 2

(see Fig. 5-4 & Fig. 5-5)

This is a cube with a hole, of constant cross-section, running through it. Again, this is successfully classified into two regions, one of which separates in the $+x$ direction and one in the $-x$ direction.

5.4.3 Test 3

(see Fig. 5-6, Fig. 5-7 & Fig. 5-8)

This is a cube with a slot out along one side. In this case the optimal solution is to produce two regions, separating in the $+y$ and $-y$ directions. In fact we produce four regions, separating in the $+z$, $-z$, $+x$ and $-x$ directions. This is because the algorithm has an in-built bias in favour of separation in the z direction. If the same object is presented in a different orientation, as in Fig. 5-8, much better results are obtained. This type of problem can be alleviated by asking the user for a preferred separation direction and orienting the model accordingly, as mentioned earlier.

5.4.4 Test 4

(see Fig. 5-9 & Fig. 5-10)

This is a cube with a hole through it and a slot in the side. It is handled reasonably well – the algorithm breaks it down into three components; although they are of slightly more irregular shape than one might produce if doing it by hand this is the minimum number of components for this object.

5.4.5 Test 5

(see Fig. 5-11 & Fig. 5-12)

This is a cube with a through hole and an internal cavity. As such, it requires the use of a core. The algorithm successfully detects this and produces a two part pattern for the mould and two for the core box. A manual solution would probably split all the elements vertically, rather than horizontally, to allow for the incorporation of core prints to retain the core, but again this reflects the in-built bias in the code in favour of separation in the $+z$ or $-z$ direction, which could be overcome simply by changing the orientation of the model. This is in many

ways the most satisfactory result – not only does the algorithm select a reasonable parting line on the object: it also produces a reasonable parting line on the core elements when the algorithm is applied to the remaining boxes in the model after those that have been succesfully classified have been eliminated (see Fig. 5-13).

5.4.6 Test 6

(see Fig. 5-14 & Fig. 5-15)

This is a slightly simplified version of a casting for a cylinder head for a single-cylinder stationary engine which is in production at Lister-Petter's Dursley site. It thus represents a test case of realistic complexity. Clearly this is not an optimal solution; large parts of the model cannot be handled, and those parts which are categorised are split in six different directions. However, the algorithm successfully processes a large proportion of the volume of the object and returns a reasonable answer for it. Problems only arise with those parts of the object where the separation directions are very heavily constrained and where there is a high degree of model complexity. This appears to be a case where, with a small amount of manual intervention, a good solution could be obtained.

5.5 Global Optimisation by Simulated Annealing

As explained in section 5.5.1, two versions of the split line selection system have been produced; one of these uses the Bath Geometric Algebra System (GAS), whereas the other is implemented in a dialect of Common Lisp on the IBM PC. The author has implemented a simulated annealing phase in both versions of the splitter software. In the GAS based version it is the primary optimisation strategy. In the PC based version, which was initially envisaged as a small scale system capable of handling simple problems, it was added at a late stage in the project

as an enhancement to the local optimisation strategy originally provided. This was an attempt to improve the handling of 'worst case' example problems. The strategy employed is to perform a local optimisation first to provide a 'reasonable' starting point and then to apply the simulated annealing phase to attempt to improve upon it. Annealing is carried out by moving boxes: in general most boxes will have more than one possible separation direction and the tactic employed is to select boxes at random and then, for each of these boxes, to select one of their permissible separation directions at random and assign the box to the list of boxes separating in that direction. This requires re-computation of the box clusters for the list from which the box has been removed and for the list to which it has been added, which makes this a moderately expensive operation in computational terms.

The initial results suggest that the annealing phase does improve performance, but at the price of significantly increased execution time. Figure 5-16 shows the results of applying the annealing phase to the object shown in Figure 5-14. Since the annealing phase uses the same separation direction classification as the local optimisation method, it has no effect on the elements of the casting that were previously classified as not separable, but on the other parts a reduction is obtained in the number of small pattern elements in the pattern set, which tends to reduce manufacturing costs.

Figure 5-2 Test Case 1 - Original Model

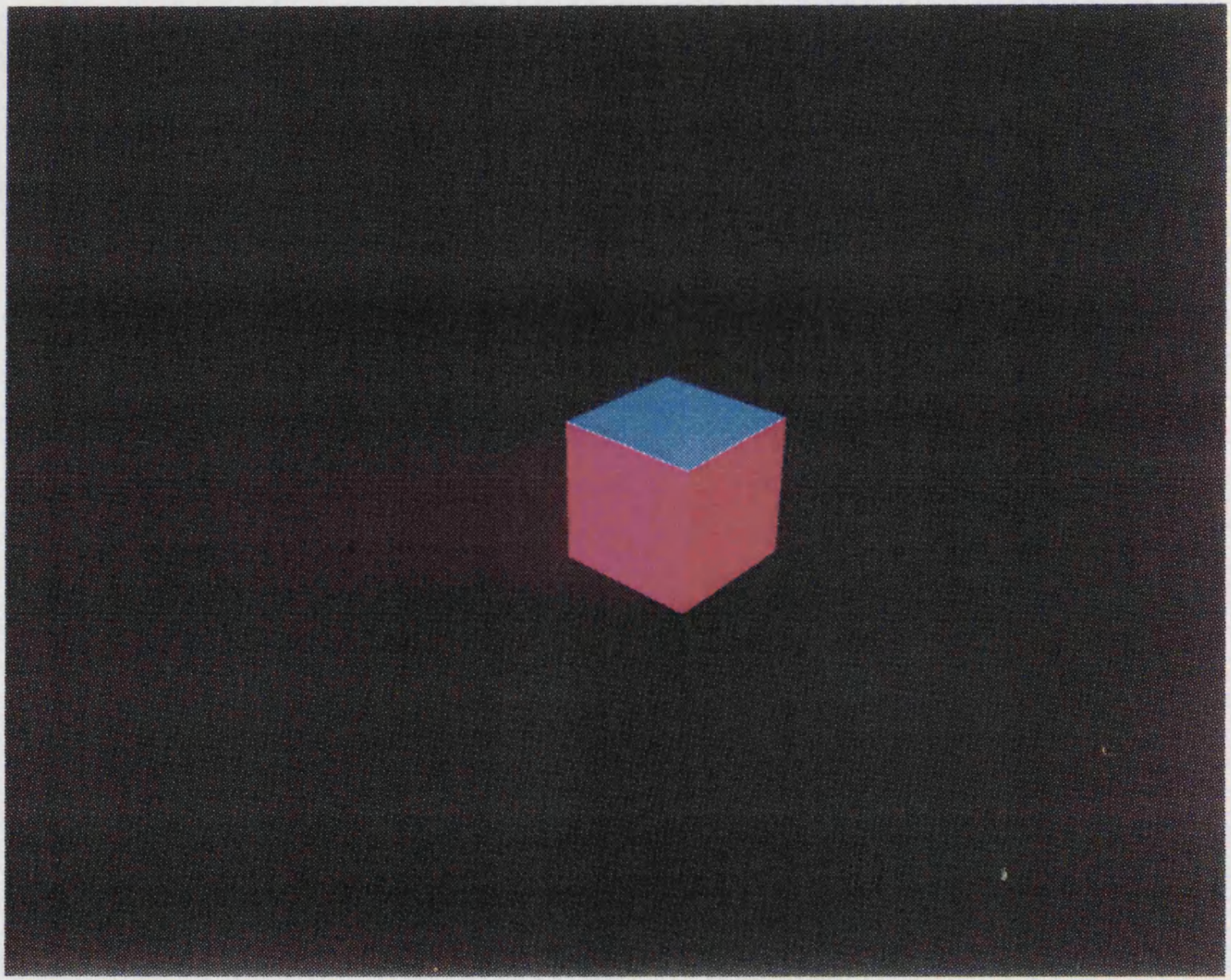


Figure 5-3 Test Case 1 - Results

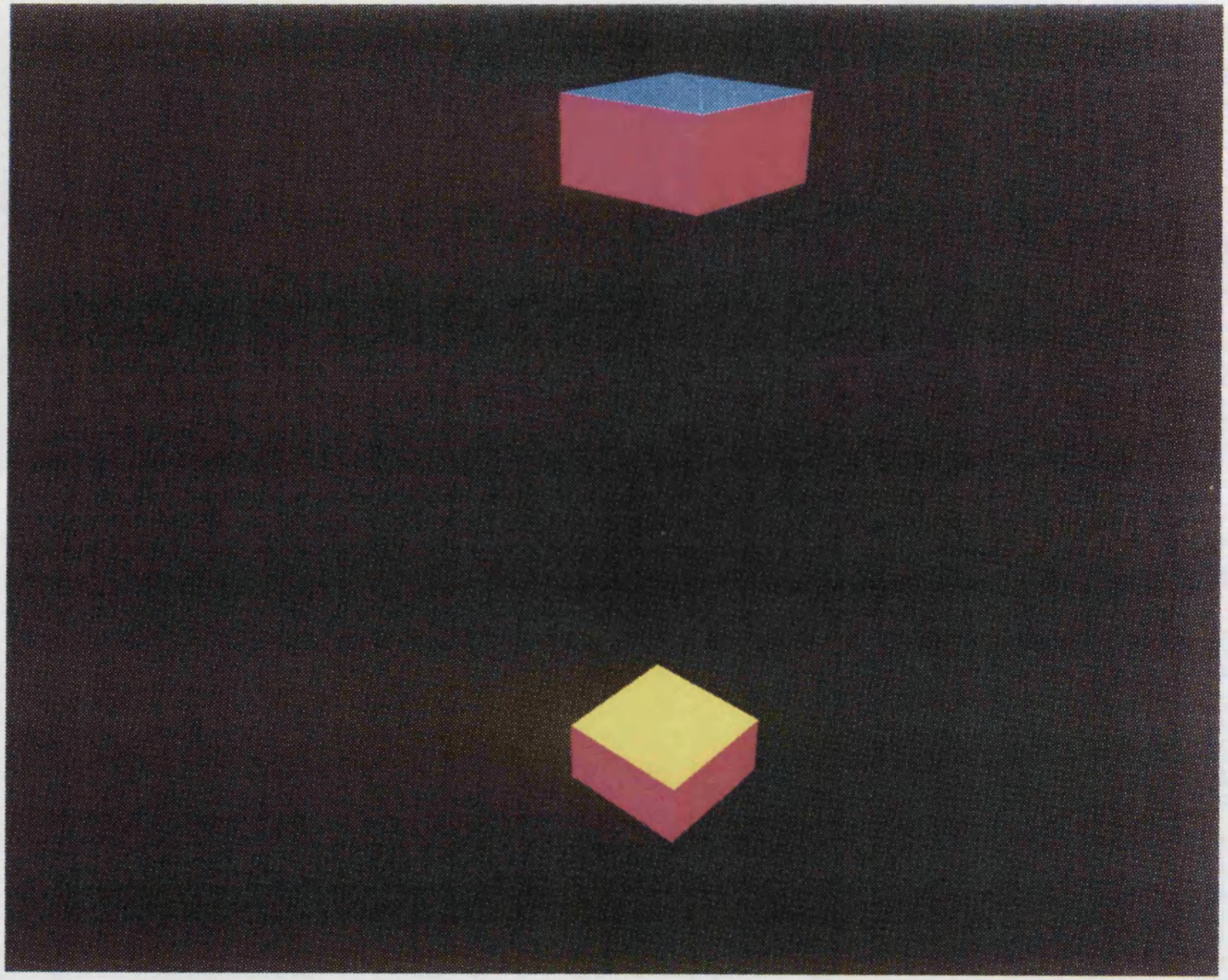


Figure 5-4 Test Case 2 - Original Model

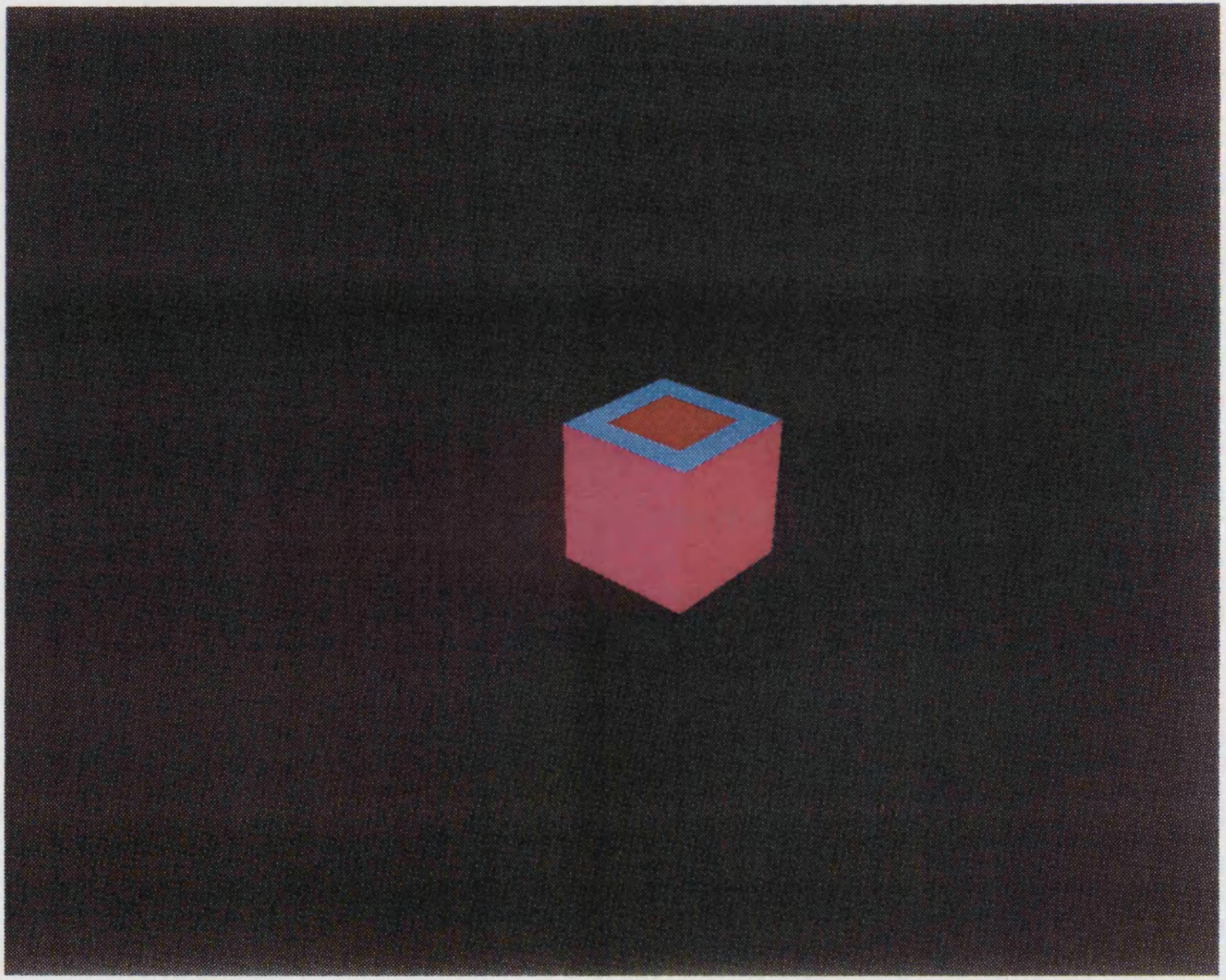


Figure 5-5 Test Case 2 - Results

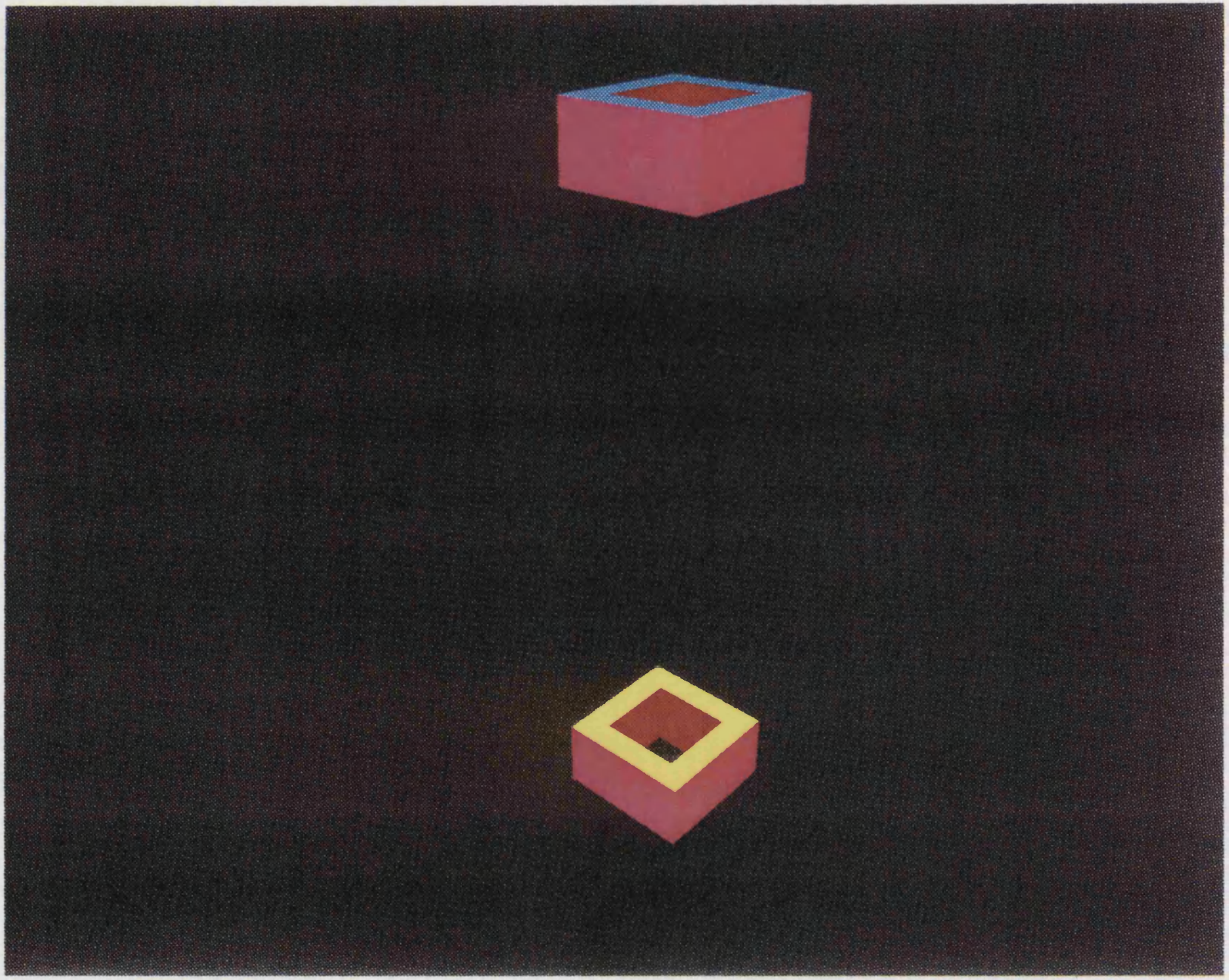


Figure 5-6 Test Case 3 - Original Model

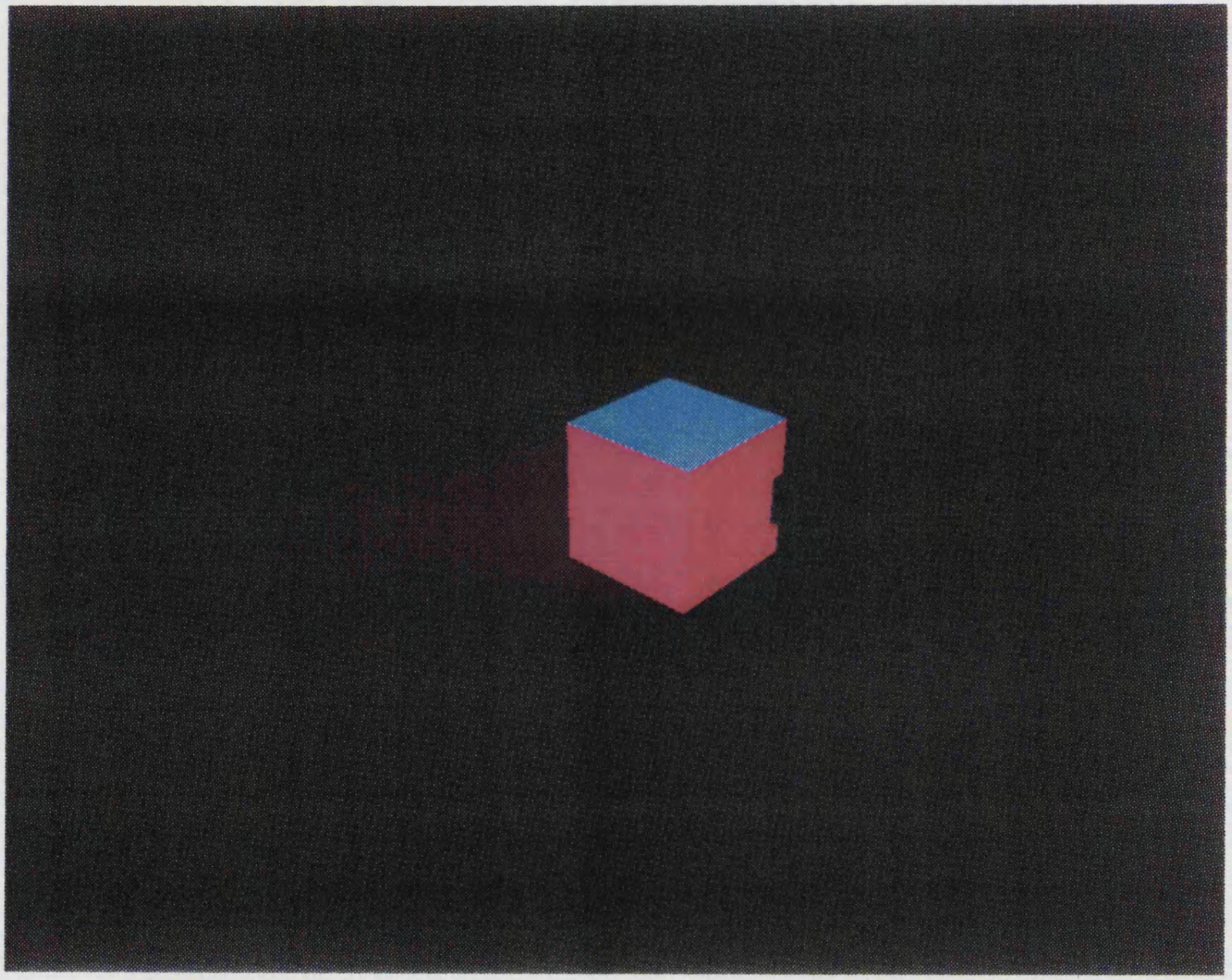


Figure 5-7 Test Case 3 - Results: Orientation 1

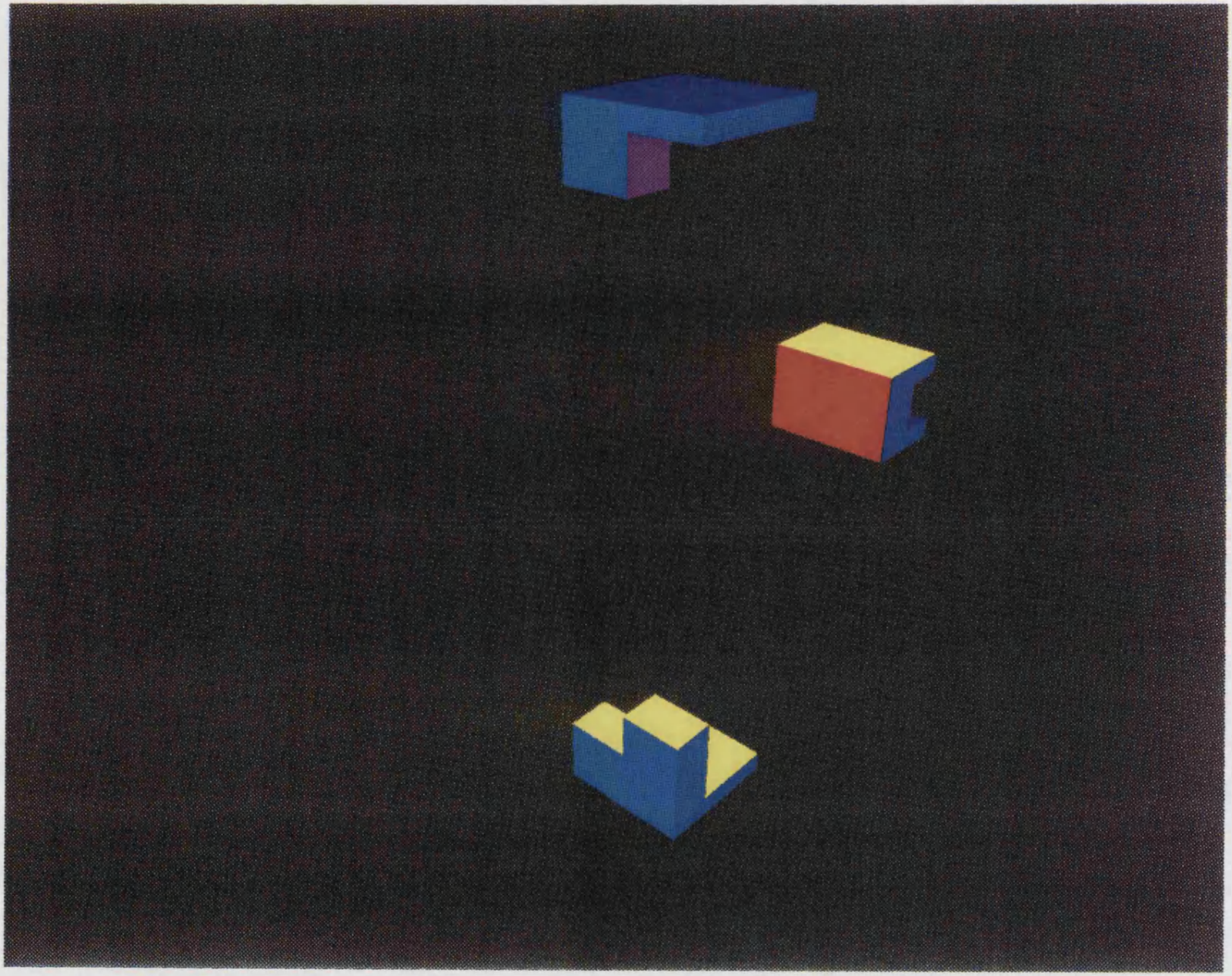


Figure 5-8 Test Case 3 - Results: Orientation 2

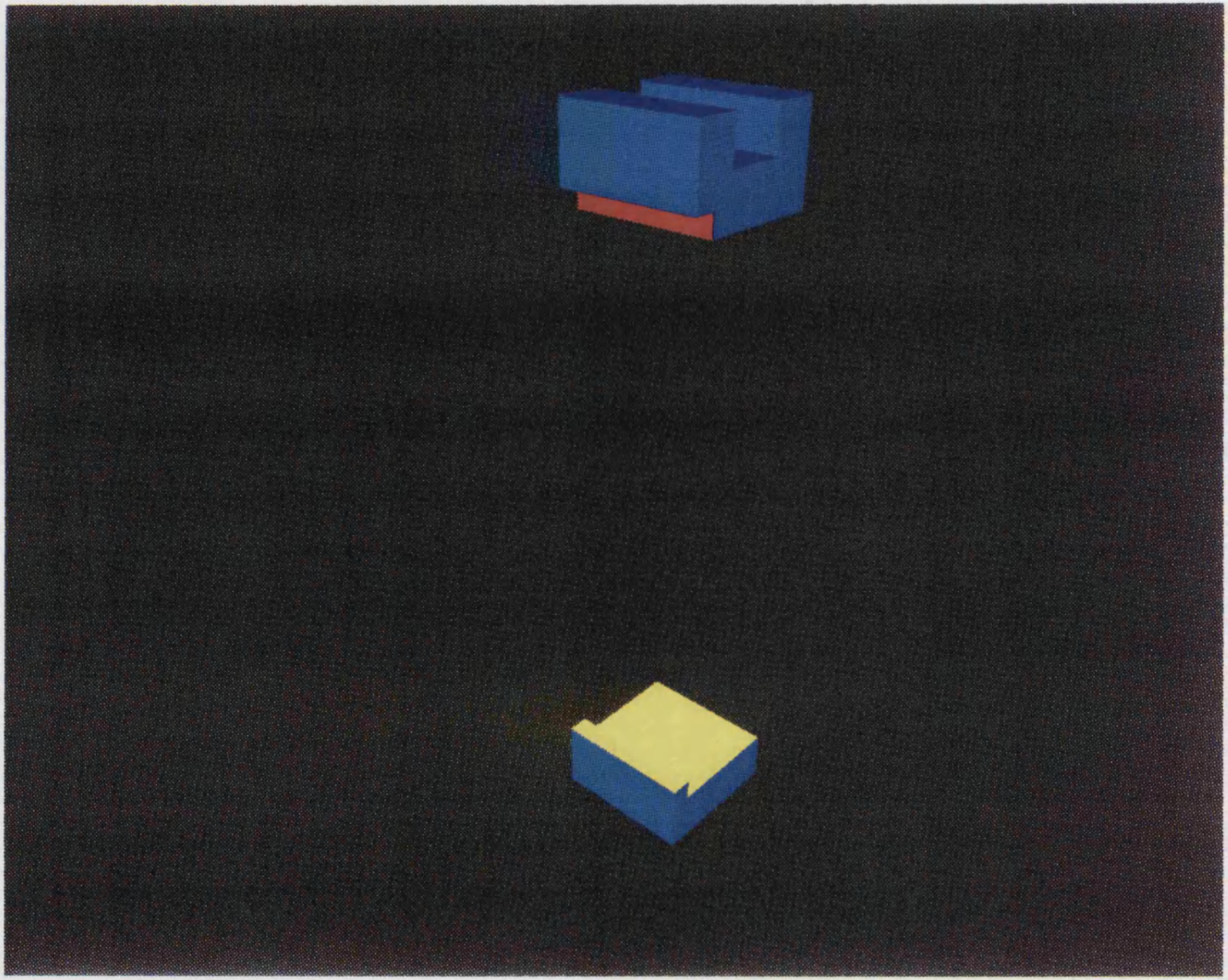


Figure 5-9 Test Case 4 - Original Model

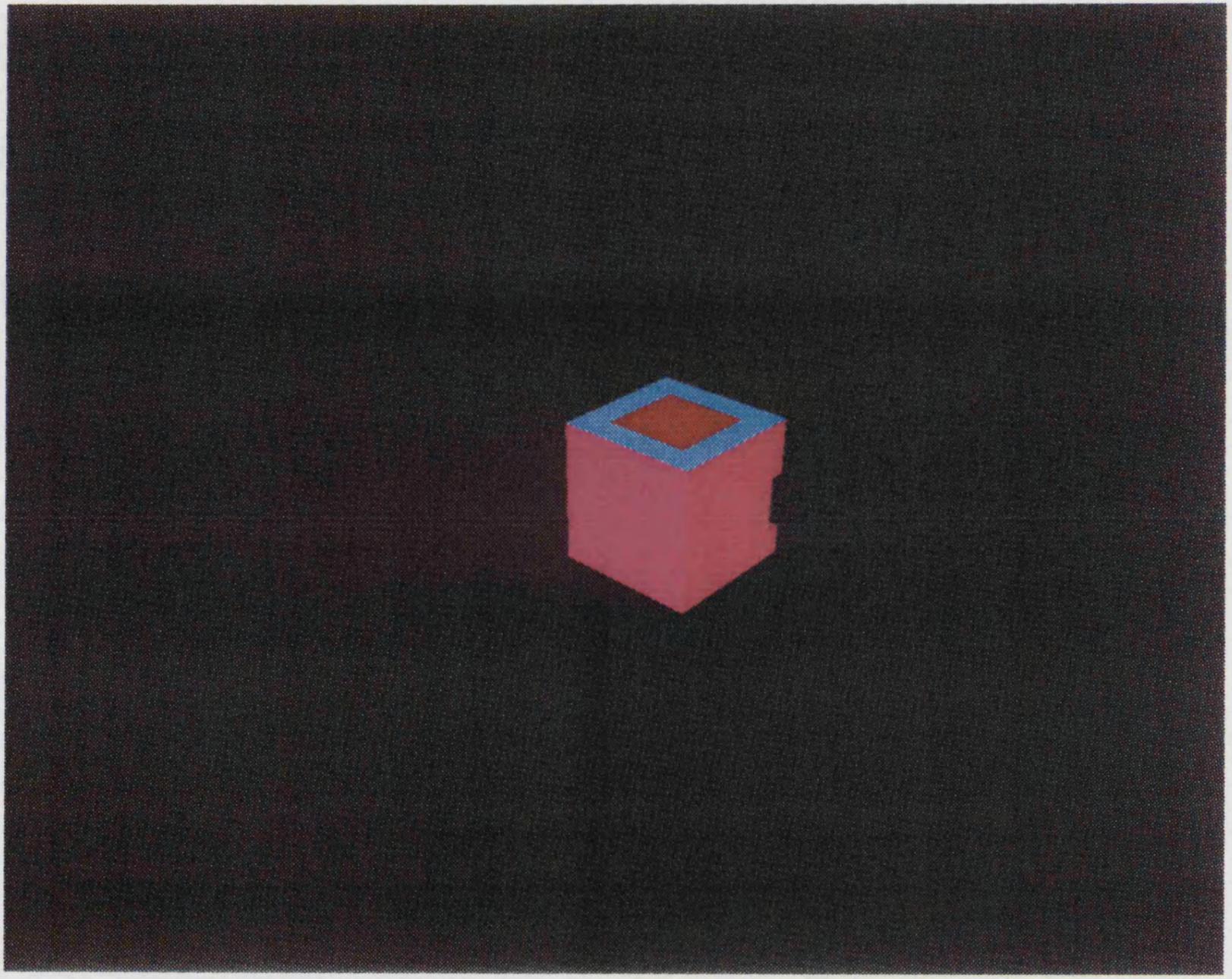


Figure 5-10 Test Case 4 - Results

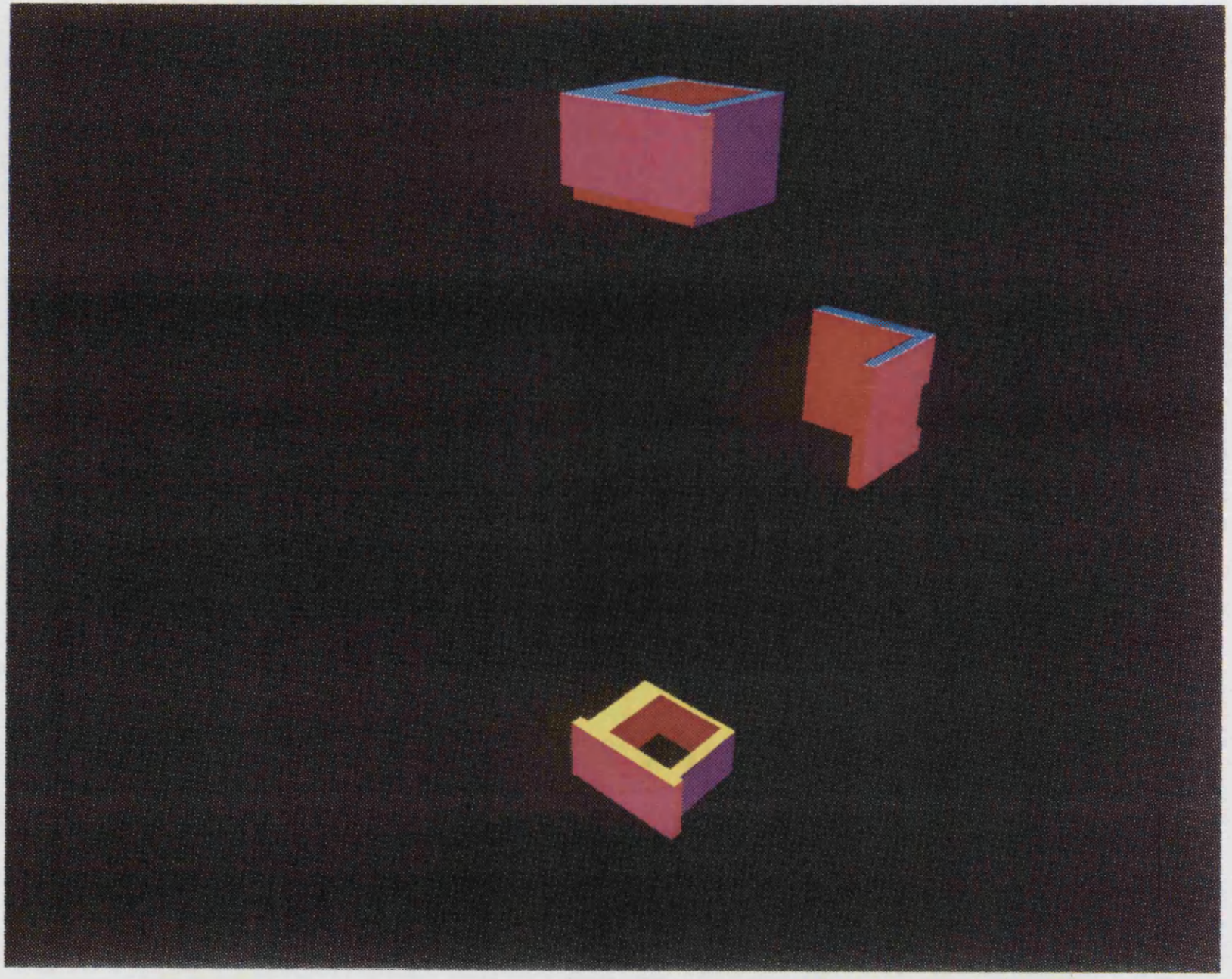


Figure 5-11 Test Case 5 - Original Model

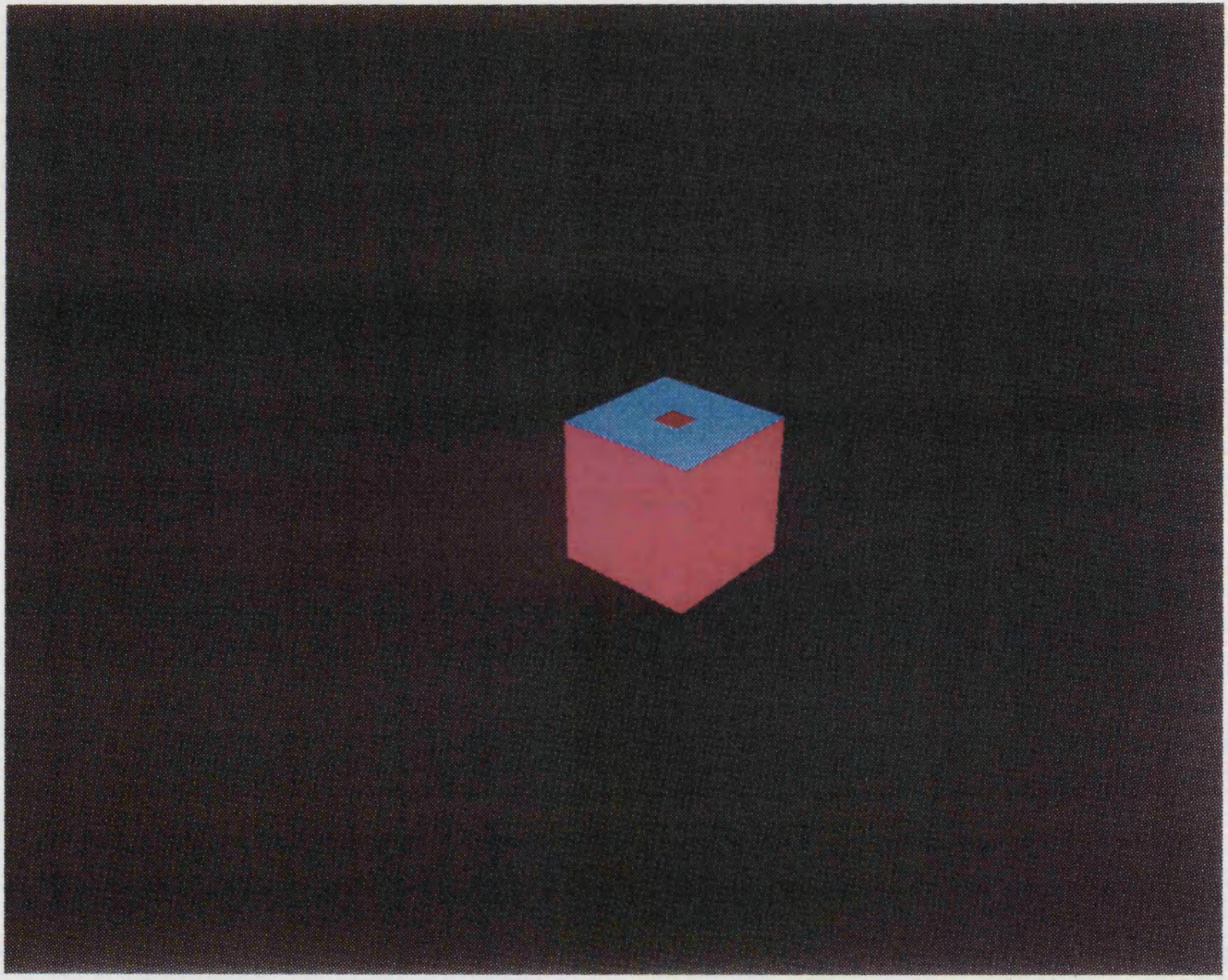


Figure 5-12 Test Case 5 - Results

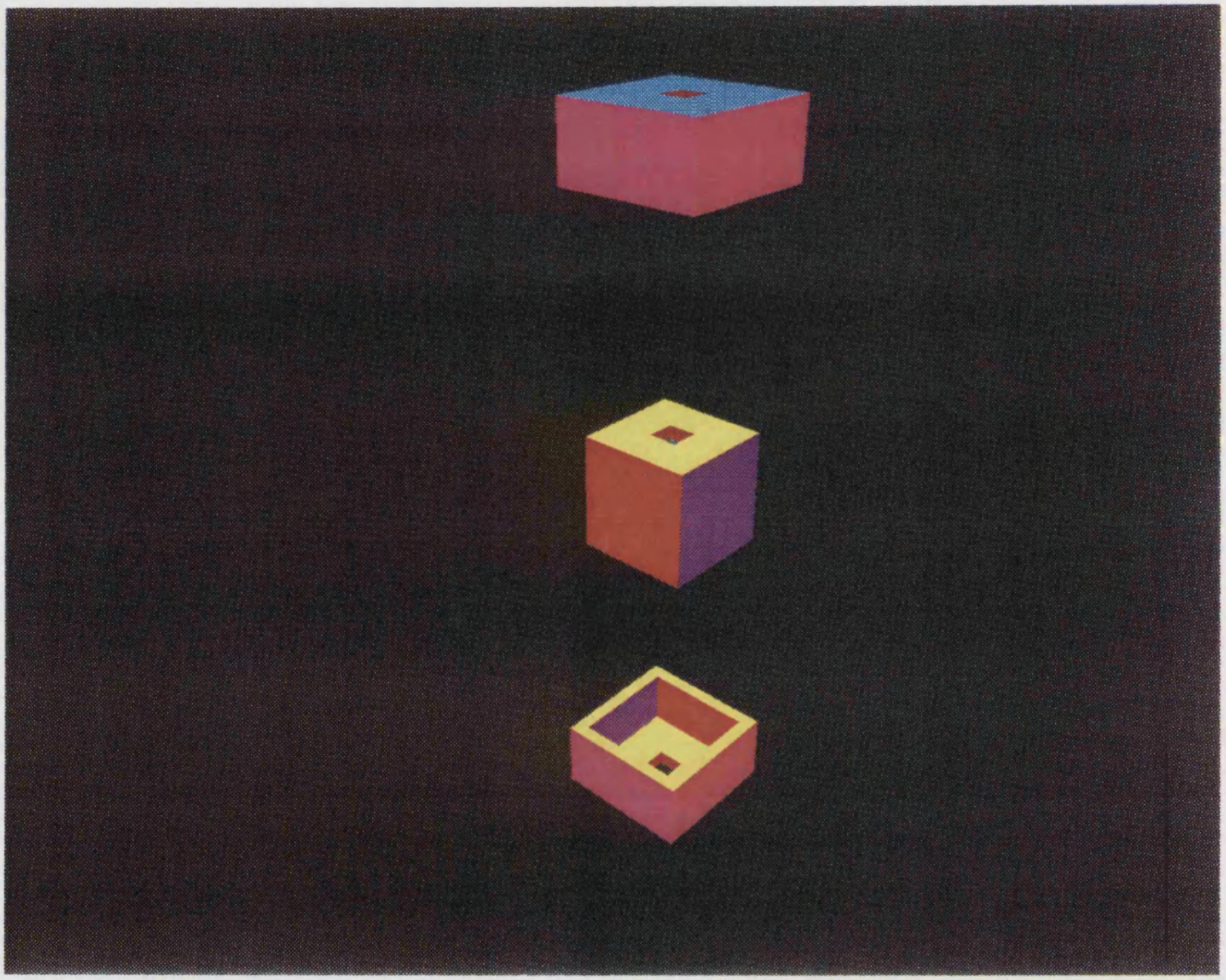


Figure 5-13 Test Case 5 - Core details

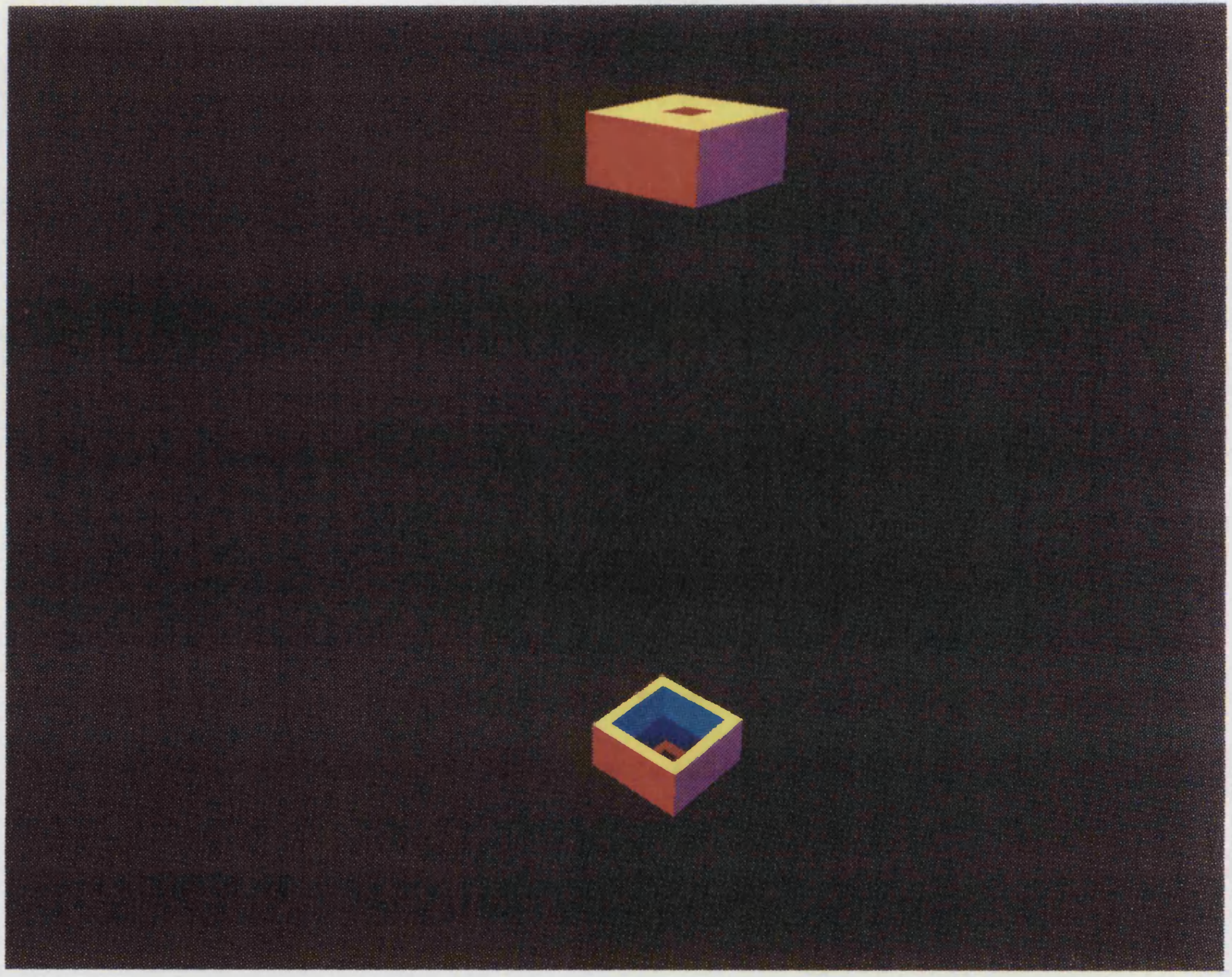


Figure 5-14 Test Case 6 - Original Model

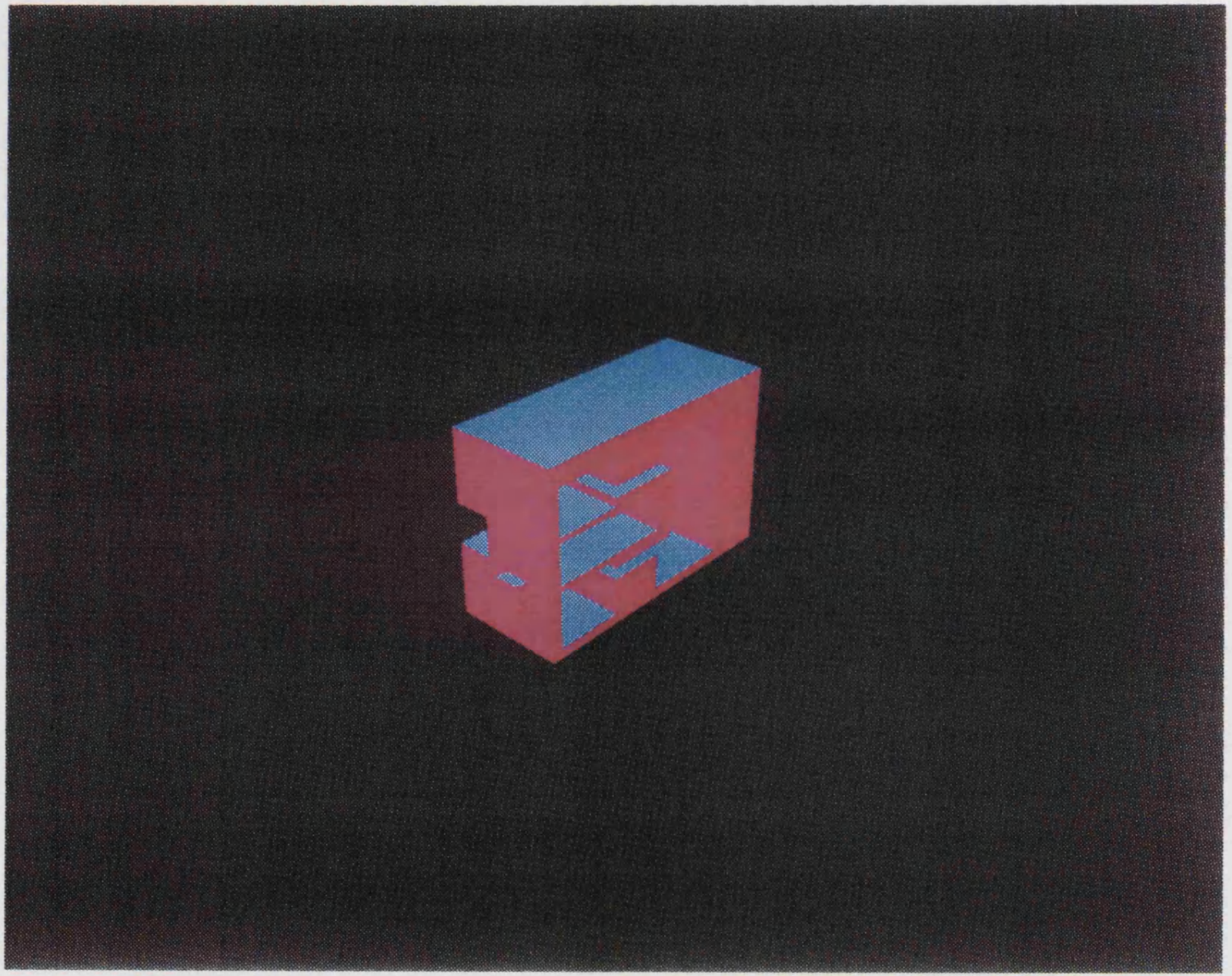


Figure 5-15 Test Case 6 - Results

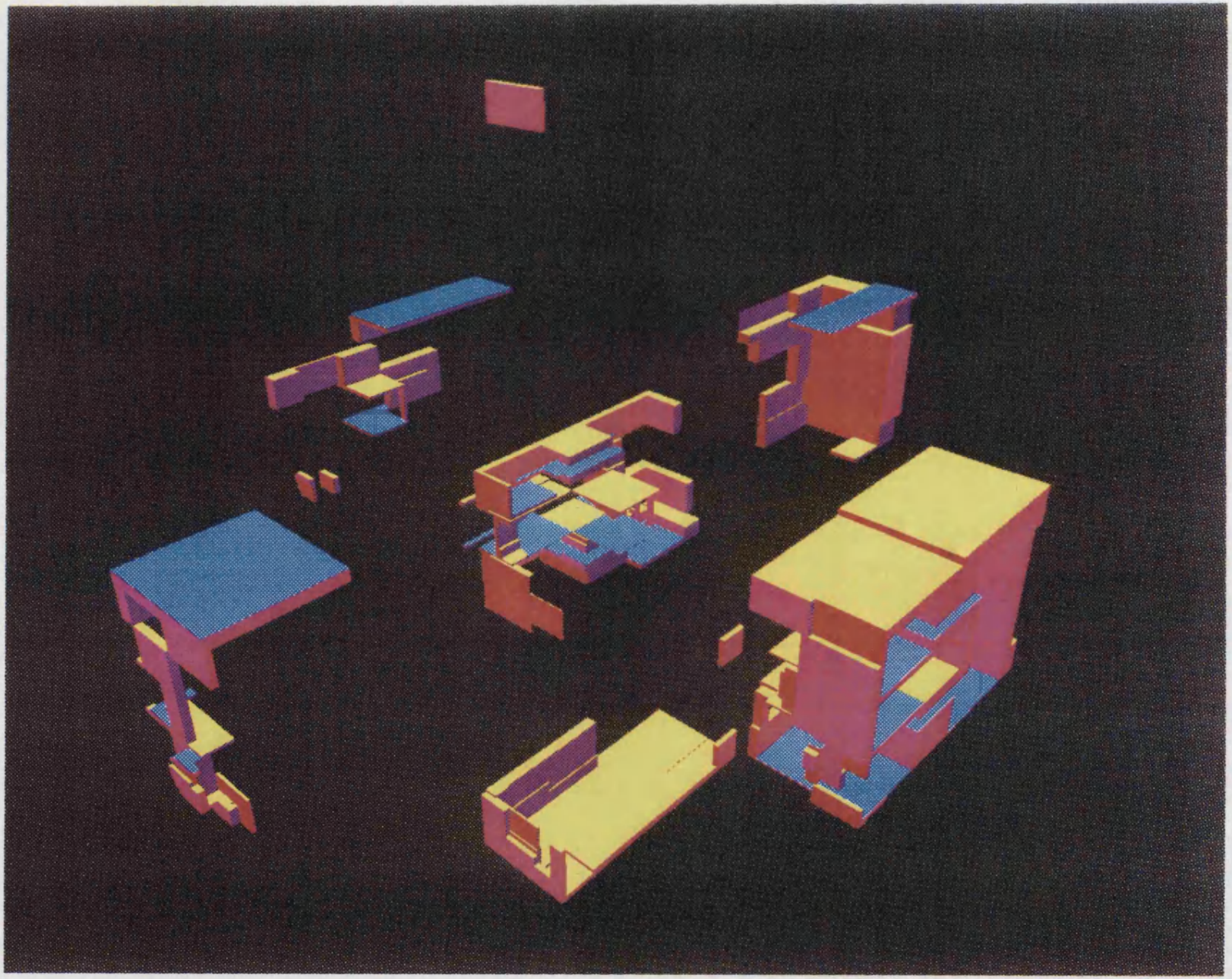
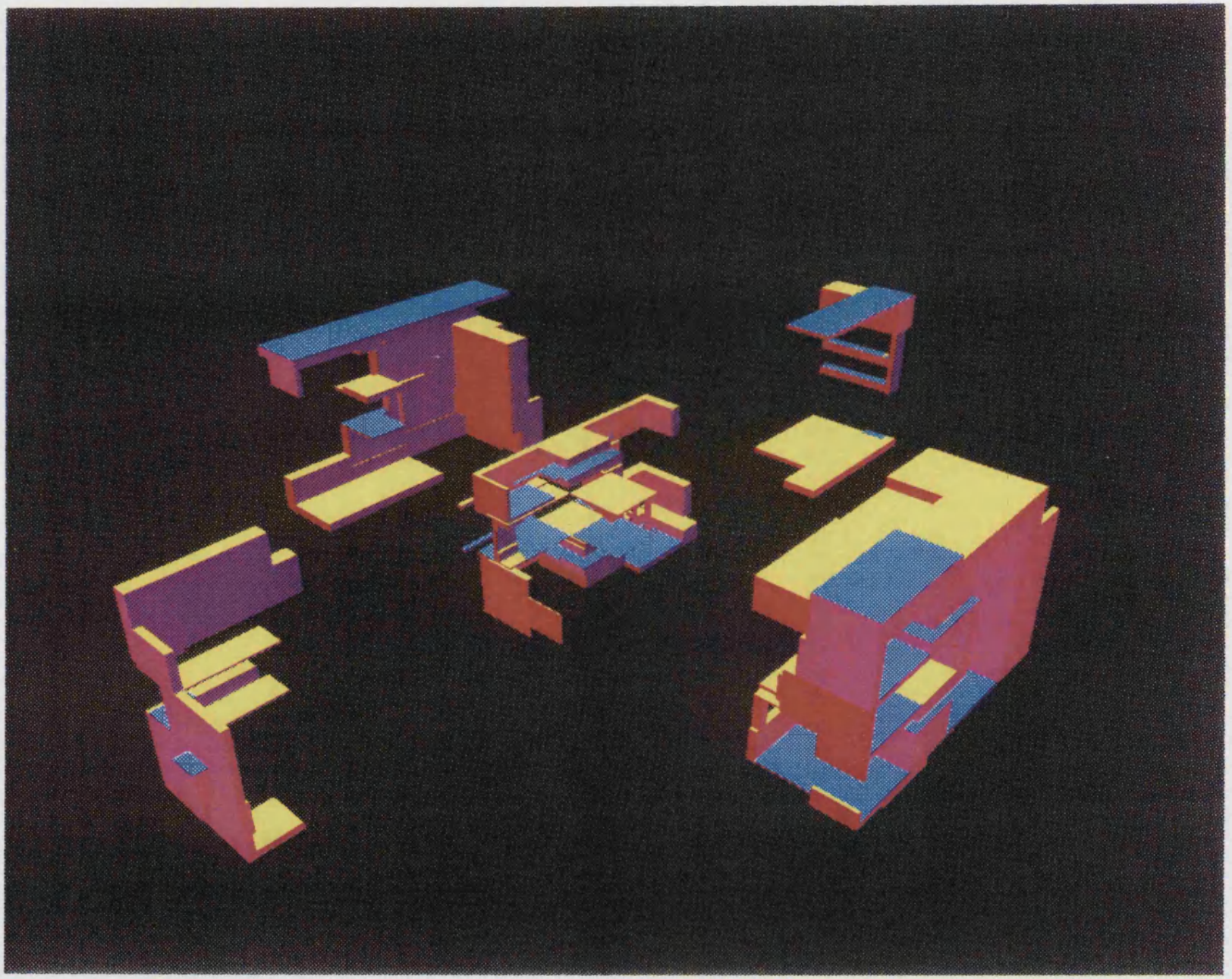


Figure 5-16 Test Case 6 - Effects of Simulated Annealing Phase



The Choice of a Cost Function

The selection of an appropriate cost function is clearly crucial to the success of the simulated annealing process. The ideal is, of course, a function which awards the best score to an optimal solution, but which also, from any given starting point, will award a better score to any modification which takes one in the direction of the optimum.

In practice such a function is unlikely to exist, and one must find a cost function which offers a reasonable compromise.

In this particular case the cost function is difficult to determine since the 'rules' for designing a good pattern are empirical and sometimes conflict with one another. It is generally accepted that one wants the smallest possible number of parts in the pattern set, and that the mould elements produced from it should all be of reasonable size – small bits are harder to handle and easier to omit inadvertently. It is also the case that additional elements in the pattern set add greatly to the cost of manufacturing and using the pattern set. Thus the best design is probably the one that uses the smallest possible number of pattern elements, provided that the elements themselves are all of a reasonable size.

A number of possible cost functions have been explored by the author, but the most satisfactory results were obtained by concentrating on attempts to minimise the number of different pattern elements. The most succesful strategy so far has been to minimise the term:

$$N_l$$

Where N_l is the number of clusters in the tentative design. There is a great deal of scope for work on improving the cost function used by the simulated annealing phase – this is an area which the author was unable to explore fully due to time constraints. The problem with this cost function is that, if one is

producing a reasonably good pattern set, then the total number of clusters, which corresponds to the number of elements in the pattern set, can be expected to be a small number. If this is the case, then the cost function will tend to behave in an unsatisfactory manner: most movements of boxes from one cluster to another will have no effect on the overall score, but occasionally moving a box will cause two clusters to be merged into one or one to be split into two. This will give a large change in the overall score. As a result of this it is very important to choose the cooling schedule for the simulated annealing operation with great care, as, with a cost function which exhibits such sharp transitions, it is very easy for the algorithm to become trapped in a local minimum. This cost function also fails to distinguish between designs producing very small mould elements and those where all the elements are of more reasonable size. A function which took account of the sizes the clusters in the design, possibly by computing minimum enclosing volumes for each cluster and penalising designs which produced small clusters, would almost certainly reduce the execution time of the algorithm (since a smoother cost function would be less demanding of a very slow cooling schedule) and increase the quality of the resulting designs. This is an area where the author would strongly recommend futher work be carried out.

5.5.1 Implementation Details

Early development work on the splitter system was carried out using the prototype implementation of the Bath Geometric Algebra System, described elsewhere [38]. This produced a system which was theoretically capable of handling arbitrary polynomial primitives but which was, unfortunately, rather slow, requiring several hours of CPU time to handle even comparatively simple test cases. At this stage the GAS system was not sufficiently stable to be able to carry out long calculation, due to, amongst other factors, development work being carried out on the garbage collecting algorithms which it used. For this reason it was decided to produce

a second implementation, which restricted itself to handling planar primitives. This simplified the code considerably and reduced the time taken to perform calculations, allowing fairly complex objects to be processed in a few hours on a moderately fast workstation (a 16 MHz Sun 4/260). This also allowed the system to be ported to other Lisp systems, without requiring the special polynomial handling capabilities provided by GAS, which allowed for the production of a version of the system which would run on David Betz's XLISP implementation of a subset of Common Lisp. This latter version has been further developed to the point where, on a machine with an 80386 CPU and a DOS extender it is capable of handling any planar problem which the GAS based version can handle. Much of the development effort has been concentrated on the PC version of the software, with the result that it now tends to produce superior solutions to the GAS version on planar models; porting the modifications back into the GAS version would, however, be fairly straightforward.

The initial *seed position* is obtained by sorting the list of boxes based on the number of directions in which each box will separate, with those with fewest separation directions being placed first. Seven lists are then constructed – one for each possible separation direction and one for boxes which will not separate. Boxes are then placed on these lists, and grouped into clusters of adjacent boxes, by adding each box to whichever of the available lists has the largest number of boxes on it already, or to the first available list (an *available* list being one of those which appears in the list of separation directions for the current box). This phase tends to produce reasonably good solutions for objects of low to moderate complexity, but can produce some very small clusters on complicated objects. The subsequent simulated annealing phase tends to iron out the difference in size between clusters.

Processing the boxes that have only one possible separation direction first ensures that the algorithm is biased towards building upon those starting points

which it is constrained to use.

A possible improvement would be to put new boxes which do not touch a cluster onto the list where they are nearest to a cluster, since this offers (presumably) a better prospect for subsequent coalescence of the resulting two clusters – this should, however, be unnecessary if a subsequent simulated annealing pass is employed

Chapter 6

An Interactive Gating Design System

6.1 Introduction

The author has developed a piece of software to allow a designer to fit a gating system onto a casting pattern. The user interface consists of a series of images of the pattern, at which the operator is able to point by means of a mouse to indicate particular areas of interest.

6.2 An Overview of the Software

The software consists of four basic modules:

- A routine which provides a visual representation of the pattern as it currently appears.
- A mouse or pointer based interface that allows the operator to select points on the picture of the pattern to which runners, risers, ingates and feeder blocks are to be attached

- A raycaster, which takes the mouse coordinates supplied by the user and translates them into a point on a surface in the three dimensional object space of the model
- A model generator, which takes the description of the desired gating system which the operator has produced by pointing at the model and translates it into a solid model in a form suitable for input to the DODO and DORA modelling systems.

6.3 User Interface

The software is designed to be simple to operate. To this end, it uses a menu-driven system to select the type of feature to be added to the pattern, currently selected from the following options:

- Riser, perpendicular to the surface to which it is attached
- Runner, attached to some surface
- Ingate
- Feeder block, cylindrical, of user specified size, perpendicular to the surface to which it is attached.

These features are defined in terms of an attachment point – a location on the pattern, specified by using the mouse, at which the feature is to be placed. The surface on which this point lies is used as a reference plane to determine the direction in which features such as risers will rise. This imposes some restrictions on the orientation of such features, but this does not appear to be a serious limitation in practice and it simplifies some aspects of the user interface considerably.

6.4 Implementational Details

The gating design system was constructed by modifying two existing programs and incorporating additional features specific to the problem. The first program is a version of the DODO system, modified for debugging purposes. This program accepts information about an object, the position from which it is viewed and the size of image to be produced, then prompts the user for x and y coordinates within the virtual image and performs raycasting operation for the specified point, returning information about whether a surface was hit, and, if so, which surface it was.

The second program is that which is normally used to display pictures produced by the modelling system. It has a feature which allows the user to interrogate a picture to determine the colour of a given pixel within an image, again for debugging purposes.

The inter-process communication features of the UNIX[1] operating system were exploited to allow data to be passed back and forth between the display program and the raycaster. Minor modifications were made to the raycaster, but the majority of changes were to the display program. This was adapted to maintain a list of pattern features destined to be incorporated into the model. Provision was also made for performing insertion and deletion operations on this list, and writing out a solid model, corresponding to the elements on the list, to a file.

The output format chosen for this program was the creation of a program in the SID modelling language[71]. This choice was made for several reasons. Firstly, it allows the model to be easily incorporated into other, more complex, models. It also allows the data to be manipulated by any of the existing suite of solid modelling tools; the mass of the gating system could be computed by the SAM program[71], for example. Additionally, it gives the user the option of hand-

[1]UNIX is a trademark of AT&T

editing the output file to incorporate gating features not currently supported by the software.

The gating design system can be seen in action in Fig 6-1 – Fig 6-4. The first of these shows a typical screen display when running the gating design system and the sequence of operations involved in adding a gating system to the casting. Fig 6-6 shows the output from a short run of the system, which illustrates the structure of the SID modelling language used to describe objects to the DODO and DORA packages, and Fig. 6-5 shows a model after the attachment of a gating system using this package.

Figure 6-1 Gating Design - Initial State

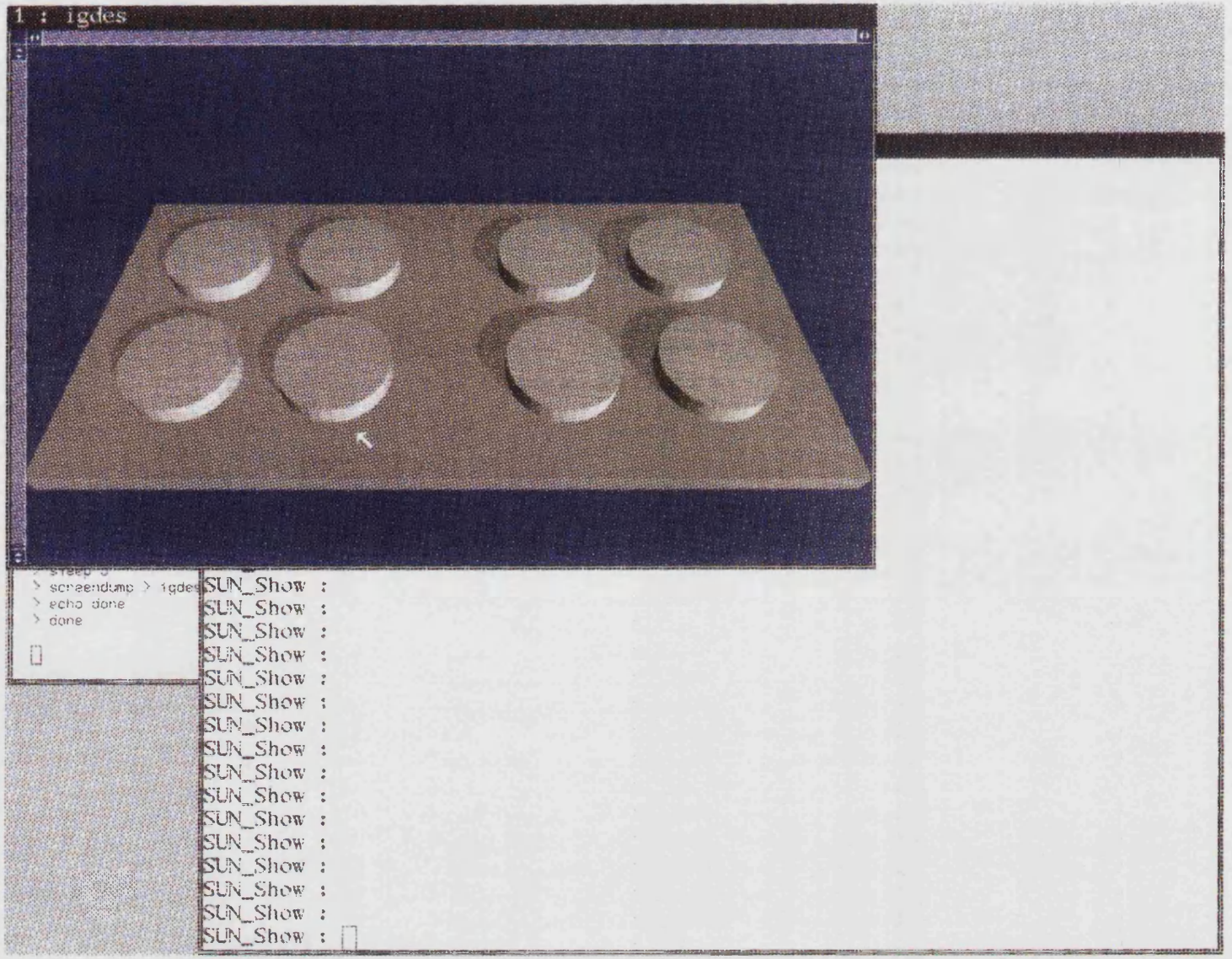


Figure 6-2 Gating Design - Creating a Runner

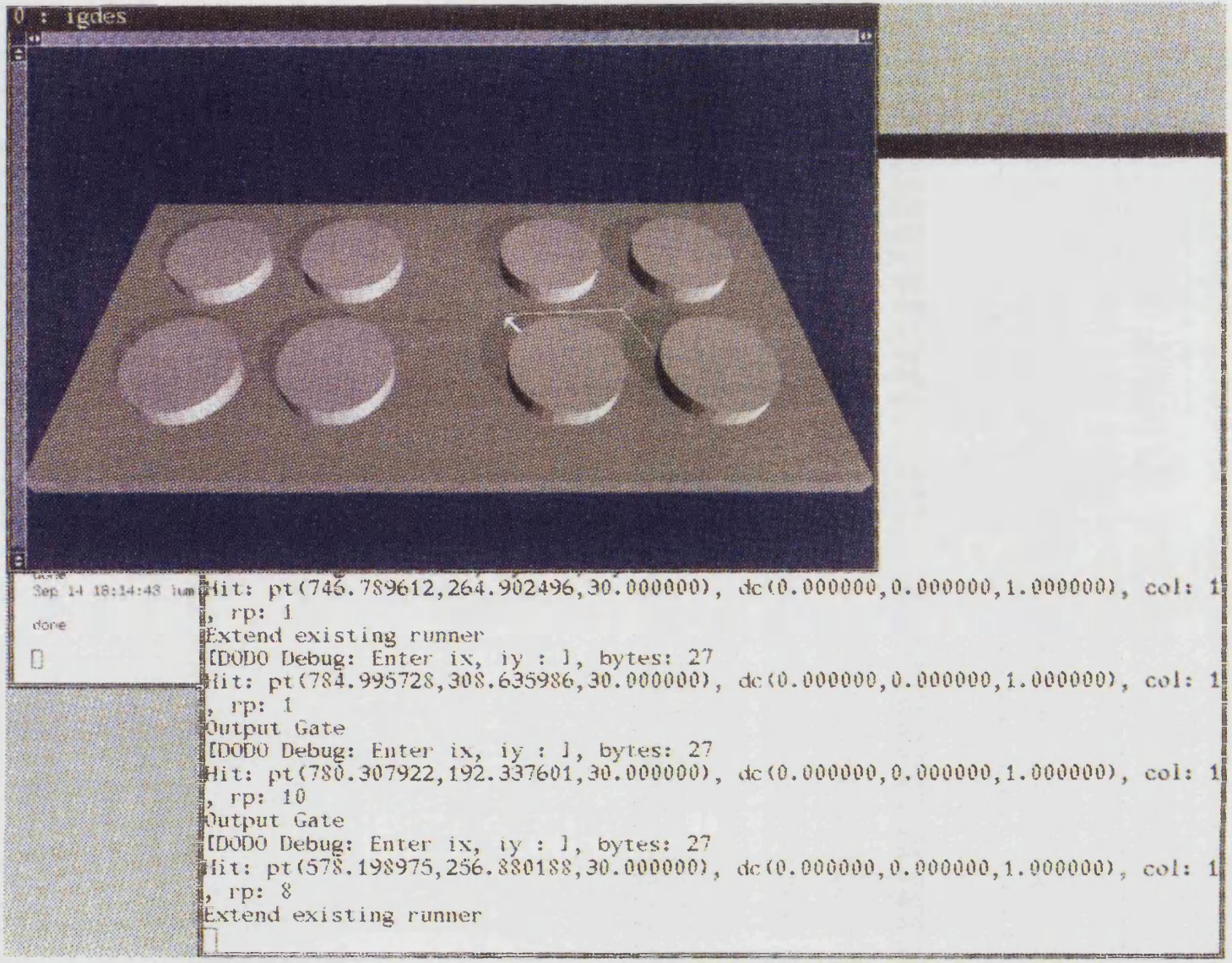


Figure 6-3 Gating Design - Adding a Riser

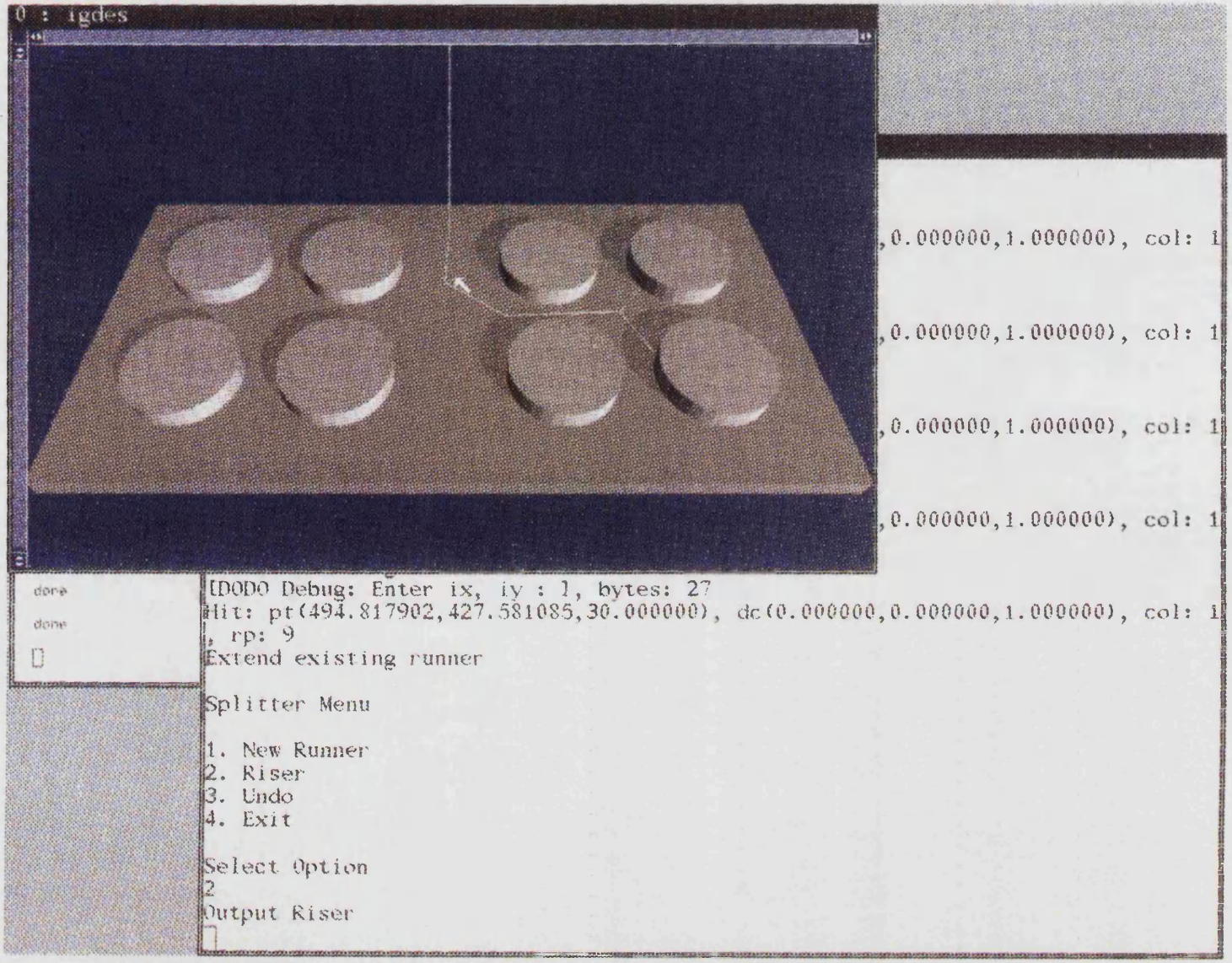


Figure 6-4 Gating Design - Completed Layout

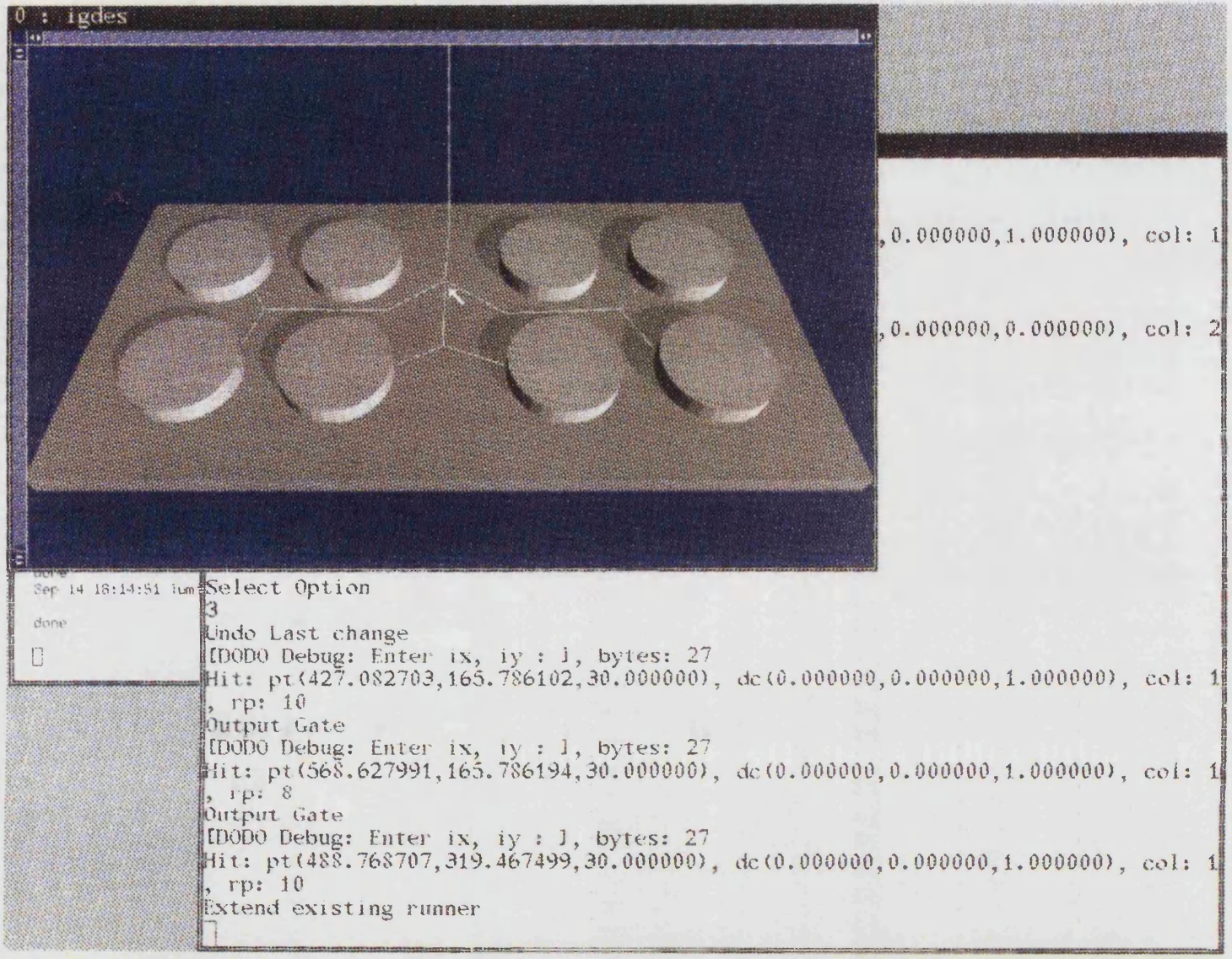
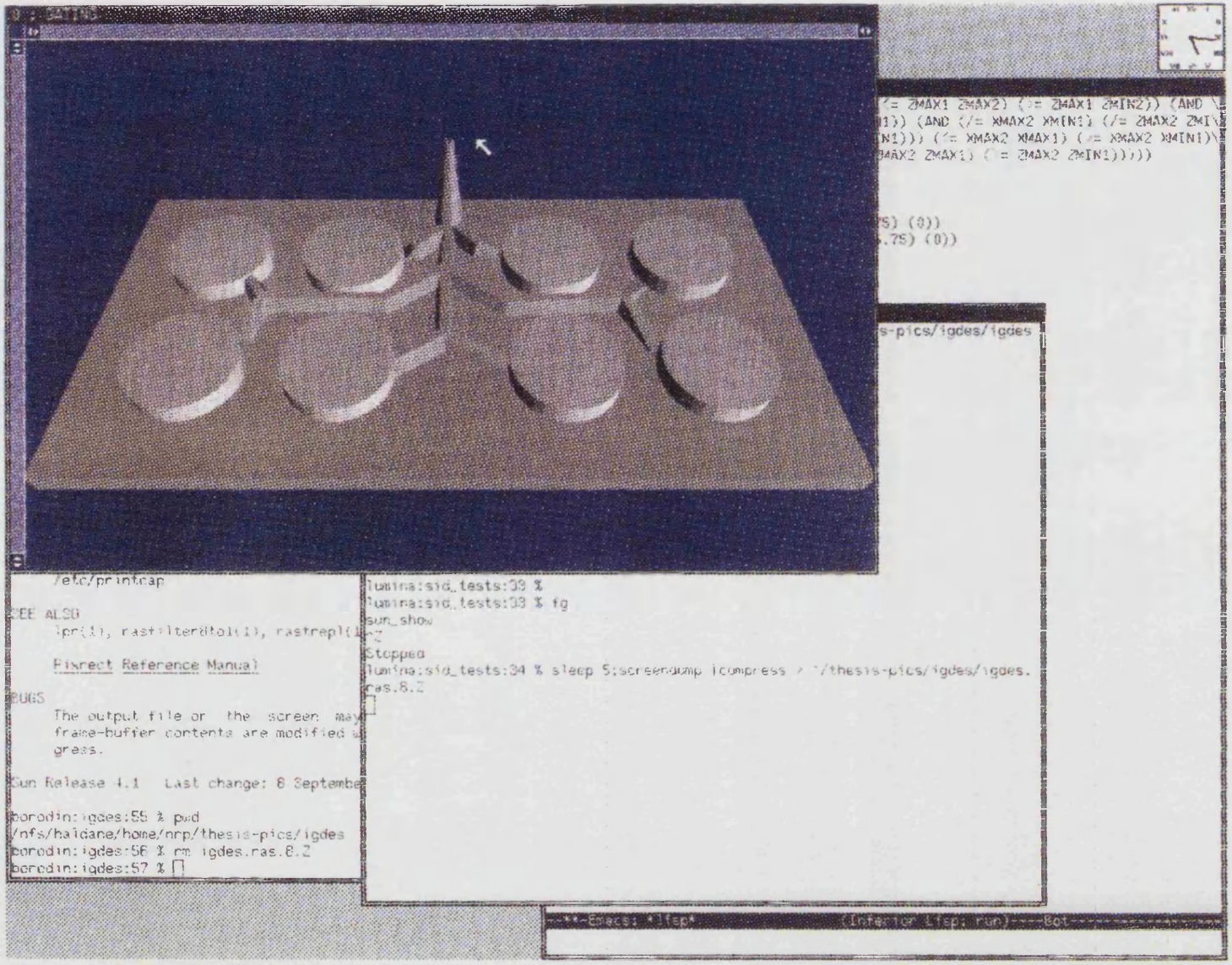


Figure 6-5 Gating Design - Results



6.5 Future Enhancements

The current version of the gating design system is seen by the author as a prototype. It needs quite a lot of work to turn it into a reliable, easy to use production system. Nevertheless, The author believes that it is worthy of further attention, since it has the potential to be a genuinely useful casting design tool.

The author also feels that a number of enhancements could be made to the kind of facilities provided by this package. Further work on these lines was precluded by lack of time, and they are offered simply as possible areas for future development.

The system could keep track of the amount of ingate and runner area needed to provide good metal flow characteristics and of the amount already available from the gating system under construction. It would also be possible to analyse the model to locate the areas with the highest moduli of solidification (see Page 25) and thus to provide the user with a list of areas where the use of feeder blocks would be beneficial, ideally with the aid of some form on 'on screen highlighting'.

The system, in its current form, restricts the object display to a single viewpoint. This can make it difficult to attach features to some parts of the casting because they are not visible in the current view. At present it is necessary to work around this by building the gating system incrementally: one fills in as much detail as possible from one viewpoint, then computes a new view of the object from a different viewpoint, and fills in the remaining detail from there. Then one merges together the two gating description files thus produced, possibly with some hand editing, to produce a complete description of the gating system. This process could be simplified greatly if the software were modified to allow the use of multiple viewpoints when selecting the location for features. This is a straightforward modification which would significantly enhance the ease of use of the system. Similar functionality is already available in the solid model measuring tool devised by Wallis[61], which inspired some features of this design.

From an aesthetic point of view, it would also be nice to improve the layout

```
sets(temp, model)
reals(scfac)
include('gating.prim.sid')
{
scfac := 1421.267090

;Autogate Gating Generation System
;file created Sat Sep 14 18:30:00 1991

;Gating details go here
;New runner
temp := addgate(pt(746.789612,264.902496,30.000000),
                pt(784.995728,308.635986,30.000000),
                pt(0.000000,0.000000,1.000000))
model := temp
temp := addgate(pt(746.789612,264.902496,30.000000),
                pt(780.307922,192.337601,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(746.789612,264.902496,30.000000),
                pt(578.198975,256.880188,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(578.198975,256.880188,30.000000),
                pt(488.751312,321.653992,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(488.751312,321.653992,30.000000),
                pt(494.817902,427.581085,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addriser(pt(494.817902,427.581085,30.000000),
                pt(494.817902,427.581085,200.000000))
model := model | temp
;New runner
temp := addgate(pt(226.988998,264.902496,30.000000),
                pt(206.114807,297.969513,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addgate(pt(226.988998,264.902496,30.000000),
                pt(209.942902,203.274796,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(226.988998,264.902496,30.000000),
                pt(203.712494,297.647003,32.727299),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(226.988998,264.902496,30.000000),
                pt(203.313202,297.247711,30.808701),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(226.988998,264.902496,30.000000),
                pt(406.051208,262.888092,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(406.051208,262.888092,30.000000),
                pt(488.751312,321.653992,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
;New runner
temp := addgate(pt(488.270996,197.783005,30.000000),
                pt(427.082703,165.786102,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addgate(pt(488.270996,197.783005,30.000000),
                pt(568.627991,165.786194,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(488.270996,197.783005,30.000000),
                pt(425.000000,156.690308,33.212002),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(488.270996,197.783005,30.000000),
                pt(488.768707,319.467499,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
;New runner
temp := addrun(pt(493.136200,417.526703,30.000000),
                pt(445.187195,369.186890,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addgate(pt(445.187195,369.186890,30.000000),
                pt(425.374786,366.851013,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
;New runner
temp := addgate(pt(541.524780,369.186798,30.000000),
                pt(567.883301,364.522614,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
temp := addrun(pt(541.524780,369.186798,30.000000),
                pt(496.562408,420.027710,30.000000),
                pt(0.000000,0.000000,1.000000))
model := model | temp
model := colour_all(model,1)
temp := read('igdes.hsp')
model := model | temp
write('igdes.gat.hsp', model)
}
```

Figure 6-6: Output from Gating Design System

of the menus provided by the system, although this is, of course, essentially a cosmetic change.

Chapter 7

Conclusions

7.1 Tapering

The author has investigated both a global tapering strategy and various schemes for localising the effects of the tapering to particular surfaces. The problem is that it is difficult to devise a localisation strategy that extends well to arbitrary (or even simply non-planar) surfaces. There are also a large number of cases where attempts to localise the tapering effect have adverse side effects. Most of these problems can be overlooked when the amount of taper applied is small and the risk of topology changes is likewise small, but in these cases it is not clear that they offer any great advantage over the simple global scaling technique, which has the advantage that it can be extended easily to handle surfaces defined by polynomials of higher degree.

7.2 Parting Line Selection

The algorithm proposed by the author is, so far as he is aware, the only parting line determination algorithm capable of handling castings with internal detail which require the use of cores. It is well behaved in that, even when it is unable to

handle the complexity of an object it returns a solution for those parts of the object which it is capable of processing. Further work is required on this system before it can be used as a practical tool, but it shows promise.

7.3 Gating Design System

The prototype of the gating design system is capable of creating gating systems and shows that the concept has potential. It does, however, need a great deal of work on the user interface in order to produce something that would be marketable. This is, however, probably the most directly exploitable part of the work carried out on this project.

Chapter 8

Recommendations for Future Work

A number of things were not investigated by the author due to lack of time. They are presented here as possible topics for other researchers to pursue.

8.1 Tapering

The current algorithms are somewhat simplistic in that they apply the same degree of taper to all surfaces on which they act. In practice one would normally apply more taper to those surfaces which had the largest surface area, and hence the largest area of contact between pattern and mould during pattern withdrawal. In the case of the localised tapering algorithm, one could approximate this behaviour by computing the area bounded by the points on a plane which was a candidate for tapering and using this information in computing the amount of taper to apply to this plane.

The application of excessive taper to an object causes changes in the topology of the object. By computing the numbers of edges, faces and vertices in an object before and after a tapering operation it should be possible to detect that this

has occurred and possibly also to detect which faces were involved. This could be used in some form of 'successive approximation' strategy to produce models which have as much taper as possible applied to them without causing topology changes.

The amount of taper required on a pattern set depends upon the moulding process and medium. The current algorithm, however, has no knowledge of casting techniques and uses a number of tuneable variables to control the tapering effect. It would be useful to incorporate some information about suitable taper angles for different applications into the system, to allow the designer to work in terms of functional and process requirements on the casting rather than having to think about the taper angles themselves.

8.2 Parting Line Selection

The current algorithm has a number of deficiencies. In particular the pruning phase does not perform a number of optimisations that would assist in simplifying the model. In particular it does not exploit the identities:

$$A \cup \bar{A} = solid$$

and

$$A \cap \bar{A} = air$$

These would be advantageous on complex models but would be expensive to apply in general during the division and pruning process. It is, however, probably worth attempting to use them to simplify boxes which have proved resistant to other methods – possibly by applying them to any box which remains unresolved after the division process has exceeded some controlling tree-depth.

The process of re-combining boxes to produce a cluster is very dependent on

the cost function used to compare two cluster lists. The current preferred strategy, after a number of alternatives were investigated, is to compute the ratios of the number of boxes in a list to the number of clusters in it, providing a reward for adding a box to the list and a penalty for creating a new cluster. More work in this area may yield an improvement in the solutions devised by the system. The real problem with this approach, however, is that it is a local, stepwise optimisation, not a global one, and so it runs the risk of finding local, rather than global, optimal solutions due to 'contour-following' effects. Post processing the results with a simulated annealing phase is an effective way of improving the quality of solutions produced by the system for complex parts, but much work is still needed to tune the annealing function to achieve reasonable results in reasonable amounts of time. The cost function used by the annealing phase also needs more attention – it makes no attempt to take into account factors such as modulii of solidification or the constraints imposed by the need to feed metal to the casting, both of which would normally be considered by an experienced patternmaker.

At a more basic level, the current cost function does not respond smoothly to small perturbations to a pattern design which take it in the direction of a better solution. This means that the simulated annealing operation needs to be carried out very slowly, with a large number of iterations of the algorithm in order to avoid becoming trapped in a local minimum. Using a more sophisticated cost function would allow the algorithm to operate with fewer iterations.

The current implementation also does not address the issue of the incorporation of core prints into the pattern set to allow cores to be located securely in position within a mould. This is a topic which would need to be addressed before a fully automated system could be produced.

8.3 Gating Design System

The work required on the gating design system is mainly development, rather than research. There are, however, one or two interesting problems in this area. One of these is the possibility of incorporating some form of expert system into the design to provide recommendations on the siting of gates and the sizes and shapes of runners and risers. Another is provision of some feedback about the modulii of solidification of various parts of the design, to allow the system to identify areas where feeder blocks would be needed, or where some redesign of the casting would be advantageous.

Bibliography

[1] *Metals Handbook – Vol 5: Forging and Casting*, American Society for Metals, ASM, 1970.

[2] *Winged Edge Polyhedron Representation*, Technical Report STAN-CS-320, Computer Science Dept, Stanford University, Palo Alto, CA, 1972.

[3] C B Besant, D P Craig, C Y Hsiung, G M J Williams, *The use of iterative CAD techniques & NC machining in mould design & manufacture*, Proc 18th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1976 pp 743-750.

[4] S K Biswas, W A Knight, *Computer aided preform design for long hot forgings*, Proc 17th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1976.

[5] A Bowyer, J H Davenport, P S Milne, J A Padget, A F Wallis, *A Geometric Algebra System*, in J R Woodwark (Ed.), *Geometric Reasoning*, Oxford University Press, 1989.

[6] I G Bralla, *Handbook of Product Design for Manufacturing*, McGraw-Hill, 1986.

[7] S A Cameron, *Modelling Space and Time* in *New Tools for Shape Modelling*, BCS Computer Graphics and Displays Group State of the Art Seminar, London, 1989.

[8] H Chiyokura, *Solid modelling with DESIGNBASE: Theory and implementation*, Addison-Wesley, 1988. ISBN 0-201-19245-4

[9] S H Choi, P Sims, T A Dean, *A complete CADCAM package for forging hammer dies*, Proc 25th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1985 pp 451-458.

[10] Clegg *Foundry CADCAM developments: an overview* The British Foundryman, October 1986, pp 397-400.

[11] A S Cramphorn, A N Bramley, *Computer aided forging design with UBET*, Proc 18th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1976 pp 717-724.

[12] M A El Dardiry, A Ibrahim, H Selim, *Family formation – a technique for reducing casting rejects*, Proc 18th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1976 pp 809-816.

[13] J H Davenport, H C Fischer, *Manipulation of Expressions* in P J L Wallis (Ed.) *Improving Floating Point Programming*, pp 149-167 Wiley, 1990.

[14] J H Davenport, *Robot motion planning*, Proc. IBM Conference on Geometric Reasoning, Winchester, England, 1986 (Oxford University Press, 1988).

[15] B J Davies, I L Darbyshire, A J Wright, *Expert systems in process planning* in A Smith(Ed) *Knowledge Engineering and expert systems in CAD*, Butterworths, 1986, pp 7-14.

[16] D C Ekey, W P Winter, *Introduction to Foundry Technology*, McGraw Hill, 1958.

[17] R A Flinn, *Fundamentals of Metal Casting*, Addison Wesley Publishing Company Inc, 1963.

[18] P J M van Laarhoven, E H L Aarts, *Simulated Annealing: theory and application*, Dordrecht: Reidel, 1987, ISBN 9-027-72513-6.

[19] Y C Lee & K S Fu, *Machine understanding of CSG: extraction & unification of manufacturing features*, IEEE Computer Graphics and Applications, 1987, Vol 7 pt 1, pp 20-32.

[20] P Gavankar, M R Henderson, *Graph based extraction of protrusions & depressions from boundary representations* Computer Aided Design Vol 22 No. 7, pp 442-450.

[21] M I Ghobrial, F Osman, A N Bramley, *Forging preform design - UBET based methods*, Proc 25th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1985 pp 465-472.

[22] M I Gokler, T A Dean, W A Knight, *Computer aided sequence design for hot upset forgings*, Proc 22nd Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1978 pp 457-466.

[23] T H Gosling, D B Welbourn, *Design for numerically controlled manufacture of patterns & dies*, Proc 17th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1976 pp 19-26.

[24] C Hoffmann, J Hopcroft, *Quadratic Blending Surfaces*, Department of Computer Science, Cornell University, April 1985.

[25] C Hoffmann, J Hopcroft, *The Potential Method for Blending Surfaces and Corners*, TR85 - 699, Department of Computer Science, Cornell University, September 1985.

[26] Hui & Tan, Mould Design using Sweep Operations, Computer Aided Design, Vol 24 No. 2, pp 81-92, February 1992.

[27] P L Jain, *Principles of Foundry Technology*, Tata McGraw Hill, 1980.

[28] D H St John, K G Davies, J G Magny, Can. Met. Quarterly, 1981, 20(3) pp 359-368.

[29] A R Johnson, D P Sturge, *The DUCT system of design & manufacture for patterns, moulds and dies*, Proc. 20th Int. Conf. on Machine Tool Design & Research, Birmingham, UK, 1979, pp 17-20.

[30] A L Johnson, R B Morris, P W Olding, D B Welbourn, *On-line designing for the foundry*, Proc 18th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1976, pp 703-710.

[31] A P Johnson, D P Sturge, *The DUCT system of design & manufacture for patterns, moulds and dies*, Proc 20th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1978 pp 17-25.

[32] H Keife, *A new technique for determination of preforms in a closed die forging of axisymmetric products*, Proc 25th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1985 pp 473-478.

[33] S Kirkpatrick, C D Gellatt Jr, M P Vecchi *Optimisation by Simulated Annealing*, Science, 220(4598), 1983, pp 671-680.

[34] P Khullar, *A Computer Aided Mould Design System for Injection Moulding of Plastics*, PhD Thesis, Cornell University, Ithaca, NY, 1981 (University Microfilms Int. 81-10935).

[35] E C Libardi, J R Dixon, M K Simmons, *Designing with features: design and analysis of extrusions as an example*, Proc. ASME Spring National Design Engineering Conference, Chicago, Il, USA, 1986.

[36] S C Luby, J R Dixon, M K Simmons, *Creating & using a features data base*, Computers in Mechanical Engineering, 1986, Vol 5 (Nov) pp 25-33.

[37] R R Martin, P C Stephenson, *Sweeping of three-dimensional objects*, Computer Aided Design, Vol 22 No. 4 pp 223-234.

[38] P S Milne *On the Algorithms and Implementation of a Geometric Algebra System*, PhD thesis, University of Bath, UK, 1990.

[39] R B Morris, D B Welbourn, *Computer graphics & numerically controlled machine tools for pattern, mould and die production*, Proc. 16th Int Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1975 pp 137-141.

[40] J Murakami, K Shirai, O Yamada, K Isoda, *A CAD system for progressive dies*, Proc 21st Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1978 pp 587-592.

[41] S Neads *Automatic Assembly of Versatile Fixtures*, PhD thesis, University of Bath, 1986.

[42] A Y C Nee, A N Poo, *Expert CAD Systems for Jigs and Fixtures*, in D T Pham(Ed.) *Artificial Intelligence in Design*, Springer-Verlag, London, 1991, ISBN 3-540-50634-9.

[43] Newman, Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, 1979, ISBN 0-07-046338-7.

[44] R D Pehlke, R E Marrone, J O Wilkes *Computer simulation of solidification*, AFS, Des Plaines, 1976.

[45] I Pillinger, P Hartley, C E N Sturgess, T A Dean, *An Intelligent Knowledge Based System for the Design of Forging Dies*, in D T Pham(Ed.) *Artificial Intelligence in Design*, Springer-Verlag, London, 1991, ISBN 3-540-50634-9.

[46] K M Quinlan *Utilising spatial locality in solid modelling*, PhD Thesis, University of Bath, 1985, BLLD microfilm No. D67117/86.

[47] B Ravi, M N Srinivasan *Decision Criteria for Computer Aided Parting Surface Design*, CAD 22(1), Butterworths, 1990.

[48] A A G Requicha, H B Voelcker, *Solid modelling: current status & research directions*, IEEE Computer Graphics and Applications, Vol 3 pt 7, pp 25-37.

[49] A A G Requicha, *Representations of rigid solids: theory, methods and systems*, Computing Surveys, Vol 12 pt 4, pp 437-464 1980.

[50] N Roberts, *CAD/CAM for the Design and Manufacture of Foundry Equipment*, pages 17-1 to 17-15, University of Warwick.

[51] R W Ruddle, Proc. *Solidification of Castings*, Inst. Metals, London, 1957.

[52] R W Ruddle, *The running & Gating of Sand Castings*, Institute of Metals monograph & report series, N°19, London, 1956.

[53] Sedgewick, Algorithms, Addison-Wesley, 1988, ISBN 0-201-06673-4.

[54] *Various papers*, Annual SIGGRAPH conferences 1980–.

[55] D P Sturge, M S Nicholls, *Developments in the DUCT system of CAE*, Proc 23th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1983 pp 157-161.

[56] W Tiller, *Rational B-Splines for Curve & Surface Representation*, IEEE CG & A, 3(6), Sept. 1983, pp 61-69

[57] S T Tan, M F Yuen, W S Sze, *Parting Lines & parting surfaces of injection moulded parts* J. Eng. Manuf. Vol 204, pp 211-220.

[58] A F Wallis, J R Woodwark, *Graphical Input to a Boolean Solid Modeller*, Proc CAD 82, Butterworths, 1982.

[59] A F Wallis, personal communication.

[60] A F Wallis, *Tool Path Verification using Set Theoretic Solid Modelling*, PhD Thesis, University of Bath, 1991.

[61] A F Wallis, J R Woodwark, *Interrogating Solid Models*, Proc. CAD 84 Conference, Brighton, England, 1984, pp 236-243.

[62] I Walker, PhD thesis, University of Bath, in preparation.

[63] P A Walsham, W A Knight, *A solid modelling approach to NC processing: some further developments*, Proc 25th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1985 pp 133-140.

[64] K K Wang, P Khullar *Computer Aided Design of Injection Moulds using TIPS-1 system*, Proc. CAM-I Int. Spring Seminar, Denver, CO, USA, 1980, pp 35-46.

[65] W P Wang, *Solid Modelling for Mould Design & Manufacture*, PhD Thesis, Cornell University, Ithaca, NY, USA, 1986.

[66] D B Welbourn, Proc. *Engineering Equipment for Foundries*, Pergamon Press, London, 1979, pp 429-435.

[67] D B Welbourn, Mater Des, 1984, 5(1), pp 7-15.

[68] D B Welbourn, G M A Cox, *Patterns and Moulds for the Automotive Industry*, Proc I Mech E, Vol 195, pages 101-109, June 1981.

[69] *Directional Solidification of Steel Castings*, New York, Pergamon Press, 1966.

[70] T C-H Woo, *Feature extraction by volume decomposition*, Proc MIT Conference on CADCAM Technology in Mech Eng, Cambridge, MA, USA, 1982 pp 76-94.

[71] J R Woodwark, A Bowyer, *Better and faster pictures from solid models*, Computer Aided Engineering Journal, Vol 3 pt 1, 1986.

[72] J R Woodwark, *Shape models in computer integrated manufacture – a review*, Computer Aided Engineering Journal, June 1988, pp 103-112.

[73] J R Woodwark, *Computing Shape*, Butterworths, 1986.

[74] J R Woodwark, *Some speculations on feature recognition*, Computer Aided Design, 1988, Vol 20 pt 4, pp 189-196.

[75] J R Woodwark, *Blends in Geometric Modelling*, IBM (UK) Scientific Centre, Winchester.

[76] G B Yu, T A Dean, *A practical computer aided approach to mould design for axisymmetric forging die cavities*, Proc 25th Int. Machine Tool Design & Research (MATADOR) Conf., Birmingham, England, 1985 pp 459-464.

[77] D Zhang, A Bowyer, *CSG Set Theoretic Solid Modelling and NC Machining of Blend Surfaces*, Proc 2nd ACM Symposium on Computational Geometry, New York, June 1986.

[78] D Zhang *CSG solid modelling and automatic NC machining of blend surfaces*, PhD thesis, University of Bath, 1986, BLLD microfilm No. DX81241.