Constrained Constructive Solid Geometry: a Unique Representation of Scenes

J.A.D.W. Anderson, G.D. Sullivan & K.D. Baker.

Intelligent Systems Group, Department of Computer Science,
University of Reading, RG6 2AX, UK.
James.Anderson@reading.ac.uk

Constraints are described for Constructive Solid Geometry which ensure that a scene composed of solids is described uniquely, up to a choice of the decomposition of compound solids into primitive ones. The constraints can be applied more easily to an Additive Constructive Solid Geometry which is also better suited to implementation on a parallel architecture.

Constructive Solid Geometry (CSG) has been used in several model-based vision programs, most notably in ACRONYM[1]. This paper examines some of the problems that arise with CSG and proposes constraints that make CSG a unique representation of scenes. It is also suggested that Additive Constructive Solid Geometry (ACSG) is more useful for vision because it leads to simpler computation of connectivity, within and between objects, and can be implemented more readily on a parallel architecture.

The motivation for examining the properties of a formal modelling system arises from what we shall call the strong thesis of model-based vision. The principal tenet is that all image features required to verify a model can be derived automatically from a model and knowledge of the optics of image formation. That is, models should provide complete knowledge of visual form which can, in principle, be used to solve any visual problem. The subsidiary tenet is that there is a two-way mapping between image locations and instantiated models. Thus, in principle, it is possible for analysis to proceed both top-down from instantiated models to the image and bottom-up from the image to instantiated models.

This latter property underlies the definition of visual knowledge as "knowledge which can be brought into a two-way, spatial mapping with an image" (compare with Sloman's similar definition[2]).

Thus, in model-based vision, models provide a justification for the particular image processing techniques used. They provide a deep knowledge of image processing and mediate between image processing and the rest of the system's knowledge. If models are to be used automatically by an intelligent vision program then it is important that the models and modelling processes are well formed and do not require human intervention.

CONSTRUCTIVE SOLID GEOMETRY

CSG is a well developed theory of geometrical modelling[3-9] which provides a complete geometrical description of objects. That is, in principle, any geometrical property of an object can be computed from a CSG[3]. This property is *necessary* to satisfy the strong modelling thesis.

CSG uses a binary tree representation whose terminal nodes are geometrical transformations or combining operations (typically union, intersection and difference). CSG necessarily allows multiple descriptions of objects because many decompositions of an object into primitive solids are possible, but previous CSGs have introduced multiplicity within the representation itself. Firstly, the parameters of the geometrical transformations have been under-constrained, which allows identical transformations to be generated with different values of the parameters and, secondly, most CSGs do not rule out the creation of null objects. This can be done in a general CSG, see Tilove[8], but is easier in an ACSG.

UNIQUE TRANSFORMATION

In the appendix, constraints are derived which guarantee that the concatenation of scale, rotation and translation is unique. The constraints restrict the range of the individual parameters of the transformations, but do not restrict the set of transformations which can be generated. That is, all of the transformations which can be generated by the unconstrained concatenation of scale, rotation and translation are generated in exactly one way by the constrained versions.

This most important result underlies the unique description of scenes in that the size, orientation and position of an object can be described in exactly one way.

COMBINATION OF SOLIDS

Tilove[8] presents algorithms for detecting null objects in CSGs. Null objects can arise in two ways. Firstly, material which is removed by difference or intersection may, later, be put back by union, or *vice versa*. This is impossible in an ACSG which uses only union. Secondly, an object can be embedded completely within another.

AVC 1988 doi:10.5244/C.2.14

This is possible in an ACSG, but can be detected quite simply. To prevent total embedding by union it is necessary and sufficient to ensure that every primitive solid has at least one point which is unique to it. This is an expensive constraint to impose, but in many vision applications it need only be checked when the models are created, not when they are being manipulated during visual search, or simulation of actions in the world.

It is useful to identify two mutually exclusive classes of union[10]. Unconnected union, where the intersection of the solids is empty, which models the placement of discrete objects in a scene and connected union, where the intersection of the solids is not empty, which models the "gluing together" of solids. The operation of gluing together two sets of unconnected objects is not well defined, so unconnected union must appear higher than connected union in a CSG tree (see Figure 1).

UNIQUE CSG

It will now be proved that a CSG represents a scene uniquely, up to a choice of primitive solids and the order they are combined in, when it uses non-null combination and the constrained transformations as shown in Figure 1.

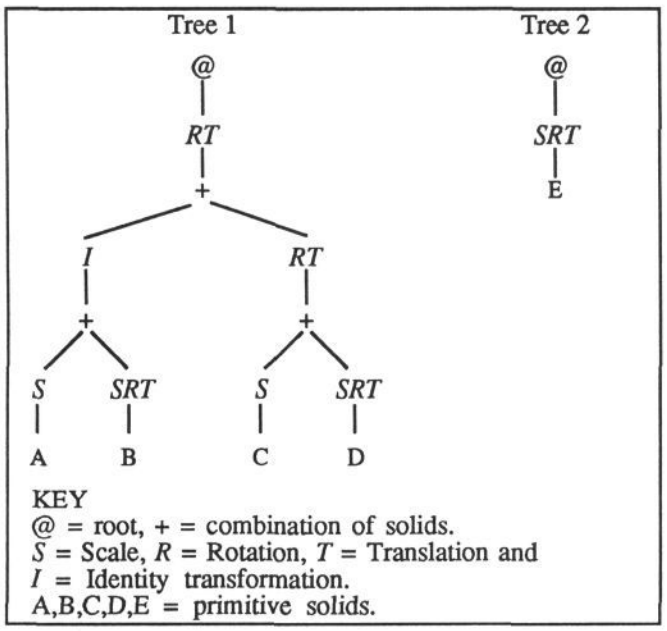

Figure 1. Unique CSG Trees

Begin at the leaves of Tree 1. A well defined primitive solid in its intrinsic coordinate frame is described uniquely. Only the terminal operators allow scaling of their children, so the size of all solids is described uniquely. Only the right hand child of a combination can be rotated or translated, so its orientation and position relative to the left hand child is described uniquely. The orientation and position of the entire scene is described relative to the left-most primitive solid. Its position and orientation in space are given by the single rotation and translation applied by the unary, root operator. Therefore the placement of objects in the entire scene is described uniquely.

If the root operator is applied to a single primitive solid then scaling is also allowed (Tree 2).

UNIQUE ACSG

ACSG inherits the uniqueness properties of CSG, but both connected union and unconnected union are transitive relations, so the order of application is unimportant, though connected union must still occur before unconnected union. Hence, an ACSG tree can be compiled down to a flat list containing the primitive solids, along with their concatenated transformations. By treating the list as a set, that is, by disregarding the order of primitives in the list, a representation is produced which is unique up to a choice of the decomposition of a scene into primitive solids. Thus the user, or meta-program, is free to chose a description suitable for the current task and, automatically, the representation of that description will be unique.

CONNECTIVITY

When an ACSG tree is compiled down to an ACSG list the connectivity of its parts can be preserved in a symmetric connectivity matrix with the rows and columns being indexed by the primitive solids' names and with the elements recording both the type of connectivity (connected or unconnected) and the discrete object to which the primitive solids belong. This may be accessed very rapidly, and in parallel, to support the maintenance of connectivity constraints.

In an ACSG tree the unconnected union operators and the root operator define discrete objects and the connected union operators define a trail of primitives solids which must be connected one to another if the entire object is to be connected. This is why a binary tree structure is retained in ACSG, without it a trail would have to be discovered at run time during the instantiation of solids.

Various levels of connectivity constraints can be checked.

1) To check that spatial interference between supposedly discrete objects does not occur, set a copy of the connectivity matrix to unconnected, then as common points are found during the instantiation mark their primitives as connected. When instantiation is finished, check that no connected primitives belong to different discrete objects.

2) To check that supposedly connected objects are, in fact, connected proceed as before, but check that all primitives connected in the original connectivity matrix are also connected in the copy.

3) To check that no primitive is totally embedded in another, instantiate the primitives and search the instantiation to ensure that there is at least one point on every connected primitive which is unique to it. In essence, this requires that a complete depth sorted list be maintained for every pixel.

In most applications it is sufficient to check the second and third constraints when models are created and only to check the first constraint, spatial interference, when they are instantiated.

Note, that checking connectivity constraints in an ACSG is simpler than in a CSG, because if any primitive is connected then the entire, discrete object is connected. This is not so in a CSG which uses intersection or difference.

SPATIAL ADDRESSING

Spatial addressing means accessing a model from a spatial location, such as a pixel in the image. This is useful in model editing to pick and place parts. It could also be used to interrogate areas of spatial interference when planning motions through a crowded environment. For example, spatial interference of a robot hand with a pliant surface might be allowed, to a certain depth, but be completely disallowed on a hard, brittle surface.

Spatial addressing might also be used in vision to discover what local, hidden features on a model could be moved in to view to explain an image feature. For example, in the bin picking task, a highly selective image feature might be noted, just outside the boundary of one well matched model, which could be explained by moving an occluded model.

The use of an ACSG simplifies spatial addressing because a point on the surfaces of a compound solid is related to just one primitive solid, whereas a point on the surface of an object constructed in a CSG, with intersection and difference, can be the product of many primitive solids.

ADVANTAGES OF ACSG

In addition to the advantages discussed earlier, there are three important computational advantages that an ACSG list and its associated connectivity matrix have over an ACSG or CSG tree.

1) Models can be instantiated by instantiating the primitives themselves, without traversing the (A)CSG tree. This is a considerable saving. For example, Plunkett[11] found that 40% of the CPU time in his parallel ray-tracing algorithm was spent traversing the CSG tree.

2) Model to model matching is simplified. After equivalent primitives in the models' ACSG lists have been deleted, any remaining primitives are the difference of one model from the other, and empty lists denote equivalence. This property might be used to discover parts which are common to models and hence support the machine learning of class hierarchies.

3) The primitive solids can be re-ordered to optimise the current computational task. So, for example, when matching modelled specularities to an image, primitives with high curvature might be scheduled firsts, or, when matching modelled texture to an image, primitives with low curvature might be scheduled first. Any ordering that carries an advantage can be used.

DISADVANTAGES OF ACSG

The major disadvantage of an additive geometry is that the primitive solids used must be very expressive to achieve adequate geometrical coverage.

It has been suggested[12] that the superquadrics are suitable primitive solids to use in vision because they are very expressive. A superquadric[12-14] is in the same form as a quadric surface, except that its exponents may take on any positive, real value. The surfaces are defined in the positive octant of Cartesian space and are extended to the other octants by reflection. These highly symmetrical solids are not suitable primitives for an additive geometry because it is seldom possible to hide unwanted parts in the body of an object. Therefore, we use only the patch of the superquadrics in one octant, the positive octant, bounded by planes.

The use of multiple primitive solids in a vision modeller, such as the family of superquadrics, raises the problem of deciding which primitive to use. With a single primitive any perceptual error by a vision system is a quantitative error in the primitive's parameters which can be corrected smoothly. We have chosen to use the supersphere octant as the single primitive solid in our modelling work. See Figure 2.

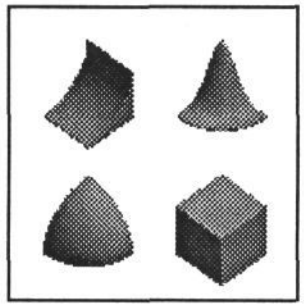

Figure 2. Supersphere Octants

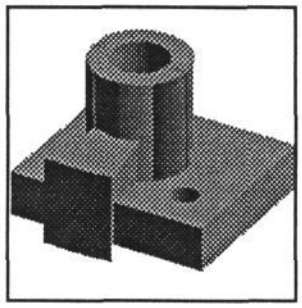

Figure 3. A "Widget"

In particular, making holes with an additive geometry is difficult. The widget, shown in Figure 3, uses four supersphere octants to make the inside of each cylindrical hole and eight to make the outside of the cylindrical annulus, though more would have given a better approximation to the surface. The entire widget is made form twenty three supersphere octants. The widget was chosen because it demonstrates these problems. An object without holes could be modelled more efficiently.

CONCLUSION

Constraints were described which make CSG a unique representation of scenes, though ACSG has many computational advantages. The proposed schemes deal naturally with parametric variation of objects, to produce classes of objects, and with articulation. All that is required is to make some of the transformations variable.

Using this modelling scheme, automatically, in a vision program would justify the strong thesis of model-based vision.

APPENDIX: TRANSFORMATIONS

The linear transformations of Euclidean 3-space can be divided into the partially overlapping sets: rotation (*R*), scale (*S*), skew and translation (*T*). In this appendix it is proved that a sub-set of these, the concatenation of scale, rotation and translation, in that order (*SRT*), can be determined uniquely by its parameters.

It is convenient to assume a right handed coordinate frame throughout and, consequently, to interpret a change of handedness as a change in the object itself, not as a change in its intrinsic coordinate frame.

It is essential that the transformations are non-singular, otherwise solids would project on to planes, lines or points. This would destroy the homogeneity of the solid representation. Rotation and translation are non-singular by definition, but scale must be constrained to non-zero values. This also prevents singularity in the concatenation of scale, rotation and translation.

Notation

Transformational matrices (T), constraints (C) and equations (E) are numbered separately in the following proofs. Standard matrix notation is used throughout describing post-multiplying, homogeneous matrices[15]. Concatenations are, however, written left to right following computer graphics convention, not right to left following mathematical convention. This is done to make the proofs more readable for those programmers who will implement the ideas presented here.

Scale

A multiplicative change of scale along the coordinate axes is given by the matrix below. *Sx*, *Sy*, *Sz* are the scale factors of their respective axes. A negative scale denotes reflection in the axis normal plane.

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(T1)}$$

Alternatively, scale may be parameterised by proportions, $Px,Py>0$, in the image plane, overall size $S>0$ and handedness in depth $H = \pm 1$. A right handed coordinate frame is used with the positive Y-axis representing increasing depth from the viewer. Thus:

$Sx = SPx, \quad Sy = SH, \quad Sz = SPz.$

It may be shown readily that either parametric description of scale is unique. A further constraint, that only one, fixed scale may be negative is derived during the derivation of constraints on *SRT*.

Rotation

Rotation about an arbitrary axis can be parameterised by the direction cosines of the axis (h,k,l) and the angle of rotation (α). Equivalently, h, k, l are the x, y, z components of a unit vector in the direction of the arbitrary axis. Following convention, rotation is reckoned positive when it is anti-clockwise as seen from the positive axis looking toward the origin. The rotation matrix, a, was derived from Paul[16] pp 25-29.

$$\begin{aligned}
a_{11} &= h^2(1 - \cos(\alpha)) + \cos(\alpha), \\
a_{12} &= hk(1 - \cos(\alpha)) + l\sin(\alpha), \\
a_{13} &= hl(1 - \cos(\alpha)) - k\sin(\alpha), \\
a_{14} &= 0, \\
a_{21} &= kh(1 - \cos(\alpha)) - l\sin(\alpha), \\
a_{22} &= k^2(1 - \cos(\alpha)) + \cos(\alpha), \\
a_{23} &= kl(1 - \cos(\alpha)) + l\sin(\alpha), \\
a_{24} &= 0, \\
a_{31} &= lh(1 - \cos(\alpha)) + k\sin(\alpha), \\
a_{32} &= lk(1 - \cos(\alpha)) - l\sin(\alpha), \\
a_{33} &= l^2(1 - \cos(\alpha)) + \cos(\alpha), \\
a_{34} &= 0, \\
a_{41} &= 0, \\
a_{42} &= 0, \\
a_{43} &= 0, \\
a_{44} &= 1. \qquad \text{(T2)}
\end{aligned}$$

Alternatively, the position of the axis can be given in spherical-polar coordinates[17], (θ,ϕ) with:

$h = \cos(\phi)\sin(\theta), \quad k = \sin(\phi)\sin(\theta), \quad l = \cos(\theta).$

It will be shown that the direction cosine parameterisation of rotation R is unique when:

1) $-\pi \le \alpha < \pi$,

2) $k \ge 0$,

3) $h \ge 0$ when $k = 0$,

4) $l = 1$ when $k = h = 0$ or $\alpha = 0$.

Equivalently the spherical-polar parameterisation of R is unique when:

1) $-\pi \le \alpha < \pi$,

2) $0 \le \phi,\theta < \pi$,

3) $\phi = 0$ when $\theta = 0$,

4) $\phi = \theta = 0$ when $\alpha = 0$.

Translation

Translation along the coordinate axes is given by the matrix below. *Tx*, *Ty*, *Tz* are the magnitudes of

translation along their respective axes. By definition, translation is unique.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix} \quad \text{(T3)}$$

Unique Parameterisation of Rotation

The derivation of a unique parameterisation of rotation given below is based on a proof of the equivalent angle and axis of rotation by Paul[16] pp 29-35.

Let $R(h_1, k_1, l_1, \alpha_1)$ and $r(h_2, k_2, l_2, \alpha_2)$ be arbitrary rotation matrices as given above (T2). It is desired to find conditions on the parameters of rotation, h, k, l, α such that $R = r$ implies that $h_1 = h_2$, $k_1 = k_2$, $l_1 = l_2$, and $\alpha_1 = \alpha_2$.

Summing the major diagonal elements gives

$$(h_1^2+k_1^2+l_1^2)(1 - \cos(\alpha_1)) + 3\cos(\alpha_1) =$$
$$(h_2^2+k_2^2+l_2^2)(1 - \cos(\alpha_2)) + 3\cos(\alpha_2),$$

but h, k, l are the components of a unit vector, therefore $h_1^2 + k_1^2 + l_1^2 = 1$. Hence

$$1 + 2\cos(\alpha_1) = 1 + 2\cos(\alpha_2)$$
$$\Rightarrow \cos(\alpha_1) = \cos(\alpha_2).$$

The cosine function is periodic, so the following constraint is chosen to restrict rotation to a single revolution

$$-\pi \le \alpha < \pi \quad \text{(C1)}$$
$$\Rightarrow \alpha_1 = \pm\alpha_2,$$
with only the negative root when $|\alpha| = \pi$ (E1)

Differencing the off-diagonal elements gives

$$[R_{23} - R_{32},\ R_{31} - R_{13},\ R_{12} - R_{21}] =$$
$$[r_{23} - r_{32},\ r_{31} - r_{13},\ r_{12} - r_{21}]$$
$$\Rightarrow [2h_1\sin(\alpha_1),\ 2k_1\sin(\alpha_1),\ 2l_1\sin(\alpha_1)]$$
$$= [2h_2\sin(\alpha_2),\ 2k_2\sin(\alpha_2),\ 2l_2\sin(\alpha_2)].$$

For $\alpha_2 \neq -\pi, 0, \pi$ the solutions are

if $\alpha_1 = \alpha_2$ then $h_1 = h_2, k_1 = k_2, l_1 = l_2$ (E2)
or if $\alpha_1 = -\alpha_2$ then $h_1 = -h_2, k_1 = -k_2, l_1 = -l_2$ (E3)

These are duplicate solutions, only one of which can be accepted. Constraints are chosen so that (E3) does not apply. Thus, choosing the constraint

$$k \ge 0 \quad \text{(C2)}$$

gives that only (E2) applies when k_1 is positive, but either of (E2) or (E3) applies when it is zero. In this case, choosing the constraint

$$h \ge 0, k = 0 \quad \text{(C3)}$$

gives that only (E2) applies, except when $k_2 = h_2 = 0$ then $l_1 = \pm 1$ and the followin constraint is chosen.

$$l = 1, k = h = 0 \quad \text{(C4)}$$

When $\alpha = \pm\pi$ the rotation matrix reduces to

$$\begin{bmatrix} 2h^2 - 1 & 2hk & 2hl & 0 \\ 2kh & 2k^2 - 1 & 2kl & 0 \\ 2lh & 2lk & 2l^2 - 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(T4)}$$

and equating diagonal terms the solutions (E2) and (E3) arise again. These are uniquely constrained as before.
When $\alpha = 0$ the rotation matrix reduces to identity and the following constraint is chosen.

$$l = 1 \text{ and } h = k = 0, \alpha = 0 \quad \text{(C5)}$$

Collecting these constraints together gives the results reported in the section "Rotation" above which is described less formally here. Rotation is uniquely parameterised when the axis of rotation lies in the Y≥0 hemisphere, but when it lies in the Y=0 plane it is constrained to lie in the X≥0 semi-disc, but when it lies along the Z-axis it is constrained to lie in the direction Z>0. Additionally, if the angle of rotation about the axis is zero, then the null rotation is performed about the Z>0 axis. Any choice of hemisphere, semi-disc and axis would do, so there are an infinite number of unique parameterisations. The one chosen here seems quite natural given a right-handed coordinate frame with the positive Y-axis representing increasing depth from the viewer at the origin.

Note that constraint (C1) implies equation (E2) which says that the inverse angle of rotation is just the negative angle, except that a half-revolution is its own inverse. This is a more natural notion of inverse rotation than, say, adding a complementary rotation in every case.

Unique Parameterisation of *SRT*

It is desired to find conditions on the parameters of the concatenation *SRT* such that $S_1R_1T_1 = S_2R_2T_2$ implies that the parameters of $S_1R_1T_1$ are equal to the

parameters of $S_2R_2T_2$. By inspection of the concatenations it can be seen immediately that $T_1 = T_2$, therefore it is only necessary to consider $S_1R_1 = S_2R_2$.

By definition scale and rotation are non-singular and therefore have unique inverses

$$\Rightarrow S_2^{-1} S_1 R_1 R_1^{-1} = S_2^{-1} S_2 R_2 R_1^{-1},$$

but rotation and scale are groups, therefore $S_2^{-1}S_1 = S_3$ and $R_2 R_1^{-1} = R_3$ where S_3 and R_3 are some scale and rotation

$$\Rightarrow S_3 = R_3.$$

Hence the off-diagonal elements of R_3 are zero

$$\Rightarrow \sin(\alpha) = 0$$

$$\Rightarrow \alpha = -\pi, 0, \pi.$$

When $\alpha = 0$ the rotation matrix reduces to the identity matrix, which is uniquely determined by the constraint (C5), but when $\alpha = \pm\pi$ the rotation matrix reduces to (T4) and it can be seen from the off-diagonal elements that at most one of h, k, l is non-zero and, because h,k,l are the components of a unit vector, the remaining value is unit, by constraint (C4). For example, when $\alpha = \pm\pi$, $h = k = 0$ and $l = 1$ the rotation matrix reduces to

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(T5)}$$

Recall that R_3 is the product of two rotations. It may be decomposed into an infinite number of pairs of rotations without invoking constraint (C4) in either transformation of the pair. Therefore, any one of h, k, l may be unit with the remaining values zero. Hence the unit value in the above matrix may appear in the position of any one of an X, Y or Z scale and the anti-unit values will occupy the other two positions. Recall also that S_3 is a product of two scales. In order to prevent two anti-unit values appearing, scale is constrained to contain at most one negative value in a fixed position. The constraint

$$S_x, S_z > 0 \text{ is chosen.} \qquad \text{(C6)}$$

This is the only additional constraint required and was given in the section, "Scale" above.

Acknowledgement

We are indebted to Anthony Worrall, of the Intelligent Systems Group, who discussed the uniqueness proofs on many occasions.

REFERENCES

1. **Brooks, R.A.** *Model-based computer vision* U.M.I. Research Press, Michigan, U.S.A. (1984).
2. **Sloman, A** "Image interpretation: the way ahead?" in *Physical and Biological Processing of Images* eds **Braddick O.J. & Sleigh A.C.** Springer-Verlag, Berlin, West Germany (1983).
3. **Requicha, A.A.G.** "Representations for rigid solids: theory, methods, and systems" *Computer Surveys* (Dec. 1980) pp 437-464.
4. **Requicha, A.A.G. & Voelker, H.B.** "Solid modelling: a historical summary and contemporary assessment" *IEEE CG&A* (March 1982) pp 9-24.
5. **Requicha, A.A.G. & Voelker, H.B.** "Boolean operations in solid modelling: boundary evaluation and merging algorithms" *PROC. IEEE* (Jan 1985) 30-44.
6. **Tilove, R.B.** "Set membership classification: a unified approach to geometric inspection problems" *IEEE Transactions on Computers* (October 1980) pp 874-883.
7. **Tilove, R.B. & Requicha, A.A.G.** "Closure of boolean operations on geometric entities" *Computer Aided Design* (Sept 1980) pp 219-220.
8. **Tilove, R.B.** "A null-object detection algorithm for constructive solid geometry" *Image Processing and Computer Vision* (July 1984) pp 684-694.
9. **Voelker, H.B. & Requicha, A.G.** "Geometric modelling of mechanical parts and processes" *Computer* (Dec 1977) pp 48-57.
10. **Hartquist, E.E. & Marisa H.A.** *UM-10/1.2 PADL-2 Users Manual* Production Automation Project, College of Engineering and Applied Sciences, The University of Rochester, Rochester, New York 14627, USA (1983).
11. **Plunkett, D.J. & Bailey, M.J.** "The vectorisation of a ray-tracing algorithm for improved execution speed" *IEEE CG&A* (Aug 1985) pp52-60.
12. **Pentland, A.P.** "Perceptual organisation and the representation of natural form" *Artificial Intelligence* Vol 28 (1986) pp 293-331.
13. **Barr, A.H.** "Superquadrics and angle preserving transformations" *IEEE CG&A* (Jan 1981) pp 11-23.
14. **Franklin, W.R. & Barr, A.H.** "Faster calculation of superquadric shapes" *IEEE CG&A* (July 1981) pp 41-47.
15. **Foley, J.D. & van Dam, A.** *Fundamentals of interactive computer graphics* Addison-Wesley, Reading, Massachusetts, U.S.A. (1982).
16. **Paul, R.P.** *Robot manipulators: mathematics, programming, and control* MIT Press, Cambridge, Massachucetts, U.S.A. (1981).
17. **Stephenson, G.** *Mathematical methods for science students,* Longman, London, U.K. (1973).