Contents

List of Algorithms

Chapter 10

Use of 3D vision

In earlier (and later) chapters, we present a constructive approach to various aspects of image processing and vision that should allow readers to reproduce ideas and build systems of their own. Most of this chapter is somewhat different; 3D vision solves complex tasks having no settled and simple theory and here we step aside and provide an overview of recent approaches, task formulations, applications and current research. We hope thereby that the reader will learn what the current state of 3D vision is; it is possible that the material herein will be at the limit of the abilities of the vision novice, but may form a useful primer for Master's or Ph.D. courses.

10.1 Shape from X

Shape from X is a generic name for techniques that aim to extracting shape from intensity images. Many of these methods estimate local surface orientation (e.g. surface normal) rather than absolute depth. If in addition to this local knowledge the depth of some particular point is known the absolute depth of all other points can be computed by integrating the surface normals along a curve on a surface [Horn 86].

Several topics belonging to this category of methods have already been mentioned, i.e. *shape from stereo* (sections 9.2.5, 9.2.11), *shape from shading* (section 9.3.3) and *photometric stereo* (section 9.3.4).

10.1.1 Shape from motion

Motion is a primary property exploited by human observers of the 3D world. The real world we see is dynamic in many respects, and the relative movement of objects in view, their translation and rotation relative to the observer, the motion of the observer relative to other static and moving objects all provide very strong clues to shape and depth – consider how just moving your head from side to side provides rich information from parallax effects. It should therefore come as no surprise to learn that attempts at shape extraction are able to make use of motion. Motion, and particularly lower level algorithms associated with its analysis, is considered in detail in Chapter 15, and in this section study is restricted to shape extraction alone.

A study of human analysis of motion is instructive, and was conducted comprehensively in a computational context by Ullman [Ullman 79]. Exactly how we make deductions from moving scenes is far from clear, and several theories have come and gone in efforts to understand this matter – in particular, *Gestaltist* theories. Gestalt psychology was a revolutionary psychological paradigm proposed in Germany in the early 20th century ('Gestalt' means 'shape' or 'form' in German). It claims that more complicated mental processes cannot be simply composed from the simpler ones, and questioned the causality of events. Its suggestion that groupings of observations are of primary importance was disproved, notably by an experiment of Ullman's. On a computer screen, he simulated two coaxial cylinders of different radii rotating about their common axis in opposite directions. The view is perpendicular to the common axis; the cylinders were not drawn, but only randomly placed dots on their surfaces. Thus what is seen (on a per-point basis), is a large number of dots moving from left to right or right to left, at varying speeds. Exactly what speed and direction depends upon which cylinder surface a dot belongs to, and at what point of rotation it is – in fact, each individual dot executes simple harmonic motion about a point that is on a line that is the projection onto the image of the axis. The striking conclusion is that the human observer is in no doubt about the nature of the scene, despite the absence of surface clues and the complete absence of structure in any single frame from the sequence.

What we exploit are particular *constraints that assist in resolving the non-uniqueness of the interpretation* of a sequence of frames as a moving 3D scene. In fact, motion may be presented to us as widely spaced (in time) discrete frames, or as (pseudo-) continuous – that is, so many frames that changes between a given pair are imperceptible. We shall examine each case separately, each using Ullman's observation that the extraction of 3D information from moving scenes can be done as a two phase process:

1. *Finding correspondences* or calculating the nature of the flow is a lower level phase that operates on pixel arrays.

2. The *shape extraction* phase follows on as a separate, higher level process. This phase is examined here.

It is worth noting that researchers are not unanimous in the view that these two phases should be held separate, and approaches exist that are different from those discussed here [Negahdaripour and Horn 85].

Note that one approach to the analysis of motion is superficially similar to that of stereo vision – images that are relatively widely separated in time are taken, and correspondences between visible features made. The solution to this correspondence problem is considered in detail in Chapter 15 and section 9.2.11. It is worth remarking here that resemblance to the stereo correspondence problem is deceptive since the scene may well contain any number of independently moving objects which could mean that correlations may be strictly local. Two images are not of the same scene, but (more probably) of the same objects in different relative positions.

Searching for correspondence in motion analysis may be easier than when attempting it in stereo imaging. It is often possible to capture a dense sequence of images (i.e. the time separation between neighboring frames is small so that corresponding features are very close, and the search for them almost trivial). Moreover, the position of the feature in the next

frame can be predicted by estimating its trajectory using techniques similar to those of control theory. The Kalman filter approach (see Section 15.4) is common.

Rigidity, and the structure from motion theorem

For now, suppose that the correspondence problem has been solved, and that it remains to extract some shape information – that is, given that a collection of points has been identified in two different views, how might they be interpreted as 3D objects? As might be expected, the large number of possible interpretations is resolved by deploying a constraint; Ullman's success in this area was based on the psycho-physical observation that the human visual system seems to assume that objects are *rigid*. This rigidity constraint prompted the proof of an elegant **structure from motion theorem** saying that *three orthographic projections of four non-coplanar points have a unique 3D interpretation as belonging to one rigid body*. We shall proceed to outline the proof of this theorem, which is constructive and therefore permits the extraction of the appropriate geometry, given point correspondences in three frames from a motion sequence. In use, the theorem allows samples of four points to be taken from an image sequence – *if* they belong to the same (rigid) body, an interpretation is generated, but if they do not, the probability of there being a chance rigid interpretation turns out to be negligibly small, meaning that the algorithm is self-verifying in the sense that it only ever generates answers that are 'correct'. Thus if there are N points in the correspondence, we might search for $\frac{N}{4}$ rigid interpretations, some of which will be invalid, and others of which will group according to the rigid object to which they belong.

The theorem proof involves a rephrasing of the problem to permit its definition as the solution of an equivalent problem in 3D geometry. Given three orthographic views of four points that have a rigid interpretation, the correspondence allows them to be labeled as O, A, B and C in each image. First note that the body's motion may be decomposed into translational and rotational movement; the former gives the movement of a fixed point with respect to the observer, and the latter relative rotation of the body (for example, about the chosen fixed point). *Translational movement*, as far as it is recognizable, is easy to identify. All that can be resolved is movement perpendicular to the projection, and this is given by the translation (in 2D) of an arbitrarily chosen point, say O. Observe that motion parallel to the projection cannot be identified.

It remains to identify *rotational motion*; to do this we can assume that O is a fixed point, and seek to identify an interpretation of A, B and C as belonging to the same rigid body as O. Accordingly, we transform the problem to that of knowing three pairs of (2D) co-ordinates for A, B and C with respect to a common origin O, each a different orthographic projection; what is now required is the (3D) directions of the projections.

Formally, suppose we have in 3D an origin O and three vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ corresponding to A, B and C; given projections of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ onto three planes Π_1, Π_2 and Π_3 of unknown orientation we require to reconstruct the 3D geometry of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. Now let the co-ordinate system of the plane Π_i be defined by vectors $\mathbf{x}_i$ and $\mathbf{y}_i$; that is, $\mathbf{x}_i$ and $\mathbf{y}_i$ are orthogonal 3D unit vectors lying in the plane Π_i. With respect to these systems, suppose that on plane Π_i the points' projections have co-ordinates (a_{xi}, a_{yi}), (b_{xi}, b_{yi}), (c_{xi}, c_{yi}) – these nine pairs are the input to the algorithm. Finally, let $\mathbf{u}_{ij}$ be a unit vector lying on the line defined by the intersection of planes Π_i and Π_j.

Elementary co-ordinate geometry gives

$$\begin{aligned} a_{xi} &= \mathbf{a}\cdot\mathbf{x}_i & a_{yi} &= \mathbf{a}\cdot\mathbf{y}_i \\ b_{xi} &= \mathbf{b}\cdot\mathbf{x}_i & b_{yi} &= \mathbf{b}\cdot\mathbf{y}_i \\ c_{xi} &= \mathbf{c}\cdot\mathbf{x}_i & c_{yi} &= \mathbf{c}\cdot\mathbf{y}_i \end{aligned} \tag{10.1}$$

Further, since $\mathbf{u}_{ij}$ lies on both Π_i and Π_j, there must exist scalars $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ such that

$$\begin{aligned} \alpha_{ij}^2 + \beta_{ij}^2 &= 1 \\ \gamma_{ij}^2 + \delta_{ij}^2 &= 1 \end{aligned} \tag{10.2}$$

and

$$\begin{aligned} \mathbf{u}_{ij} &= \alpha_{ij}\mathbf{x}_i + \beta_{ij}\mathbf{y}_i \\ \mathbf{u}_{ij} &= \gamma_{ij}\mathbf{x}_j + \delta_{ij}\mathbf{y}_j \end{aligned} \tag{10.3}$$

and hence

$$\alpha_{ij}\mathbf{x}_i + \beta_{ij}\mathbf{y}_i = \gamma_{ij}\mathbf{x}_j + \delta_{ij}\mathbf{y}_j \tag{10.4}$$

We can take the scalar product of this equation with each of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, and using equation (10.1) see that

$$\begin{aligned} \alpha_{ij}a_{xi} + \beta_{ij}a_{yi} &= \gamma_{ij}a_{xj} + \delta_{ij}a_{yj} \\ \alpha_{ij}b_{xi} + \beta_{ij}b_{yi} &= \gamma_{ij}b_{xj} + \delta_{ij}b_{yj} \\ \alpha_{ij}c_{xi} + \beta_{ij}c_{yi} &= \gamma_{ij}c_{xj} + \delta_{ij}c_{yj} \end{aligned} \tag{10.5}$$

– thus we have relations between unknowns $(\alpha, \beta, \gamma, \delta)$ in terms of known quantities $(a_x, a_y,$ etc.).

It is easy to show that the equations (10.5) are linearly independent (this is where the fact that O, A, B and C are not coplanar is used). Therefore, using the constraint of equation (10.2), it is possible to solve for $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ – in fact, there are two possible solutions that differ in sign only.

This (findable) solution is important as it means that we are able to express the vectors $\mathbf{u}_{ij}$ in terms of the co-ordinate basis vectors $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{x}_j$ and $\mathbf{y}_j$. To see why this is important, picture the three planes in 3D – they intersect at the common origin O and therefore define a tetrahedron; what interests us is the *relative* angles between the planes, and if the geometry of the tetrahedron can be recaptured, these angles are available. Note though that knowledge of $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ allows calculation of the distances

$$\begin{aligned} d_1 &= |\mathbf{u}_{12} - \mathbf{u}_{13}| \\ d_2 &= |\mathbf{u}_{12} - \mathbf{u}_{23}| \\ d_3 &= |\mathbf{u}_{13} - \mathbf{u}_{23}| \end{aligned} \tag{10.6}$$

For example,

$$\begin{aligned} \mathbf{u}_{12} - \mathbf{u}_{13} &= (\alpha_{12}\mathbf{x}_1 + \beta_{12}\mathbf{y}_1) - (\alpha_{13}\mathbf{x}_1 + \beta_{13}\mathbf{y}_1) \\ &= (\alpha_{12} - \alpha_{13})\mathbf{x}_1 + (\beta_{12} - \beta_{13})\mathbf{y}_1 \end{aligned} \tag{10.7}$$

and hence

$$d_1 = (\alpha_{12} - \alpha_{13})^2 + (\beta_{12} - \beta_{13})^2 \tag{10.8}$$

since $\mathbf{x}_1$ and $\mathbf{y}_1$ are orthogonal. Now the tetrahedron formed by the three intersecting planes is defined by the origin O and a triangular base – we might consider the base given by the three points at unit distance from the origin. By construction, this triangle has sides d_1, d_2, d_3, and we can thus reconstruct the required tetrahedron.

Determining the 3D structure is now possible by noting that a particular point lies at the intersection of the normals to any two of the planes drawn from the projections of the point concerned.

There is a complication in the proof not discussed here that occurs when one of the d_i is zero, and the tetrahedron is degenerate. It is possible to resolve this problem without difficulty – the full proof is given in [Ullman 79].

It is worth noting that Ullman's result is best possible in the sense that unique reconstruction of a rigid body cannot be guaranteed with fewer than three projections of four points, or with three projections of fewer than four points. It should also be remembered that the result refers to *orthographic* projection when in general image projections are *perspective* (of which, of course, the orthographic projection is a special case). This turns out not to be a problem since a similar result is available for the perspective projection [Ullman 79]. In fact this is not necessary, since it is possible to approximate neighborhoods within a perspective projection by a number of different orthographic projections; thus in such a neighborhood, the theorem as outlined is valid. Interestingly, there seems to be evidence that the human visual system uses this sort of orthographic approximation in extracting shape information from motion.

This result is of particular value in *active* vision applications [Blake and Yuille 92, Aloimonos 93] such as a robot arm having a camera mounted upon it; when such a system finds itself unable to 'see' particular objects of interest, the arm will move for a different view, that will then need reconciling with earlier ones.

Shape from optical flow

The motion presented to human observers is not that considered in the previous section, but rather is continuous – the scene in view varies smoothly. The approach of considering widely spaced (in time) views is therefore a simplification, and it is natural to ask how to treat the 'limiting case' of separate frames being temporally very close to each other – it is well known that, in fact, the human eye perceives continuous motion from relatively few frames per second (as illustrated by cinema film). Clearly the approach of making correspondences is no longer any use since corresponding points will be separated by infinitesimally small distances – it is the apparent velocity (direction and speed) of pixels that is of interest in the study of continuous motion. In a continuous sequence, we are therefore interested in the apparent movement of each pixel (x, y) which is given by the *optical flow field* $(\frac{dx}{dt}, \frac{dy}{dt})$. In Chapter 15, optical flow is considered at length, and an algorithm described for its extraction from observation of changes in the intensity function (gray levels); accordingly, in this section, it is assumed that the flow field is available, and we ask how it may be used to extract shape in the form of surface orientation (in fact optical flow is useful for deducing a number of motion properties, such as the nature of the translational or rotational movement – these points are considered in Chapter 15).

Determining shape from optical flow is mathematically non-trivial, and here an early simplification of the subject is presented as an illustration [Clocksin 80]. The simplification is in two parts;

- Motion is due to the observer traveling in a straight line through a static landscape. Without loss of generality, suppose the motion is in the direction of the z axis of a viewer centered co-ordinate system (i.e. the observer is positioned at the origin).

- Rather than being projected onto a 2D plane, the image is seen on the surface of a unit sphere, centered at the observer (a 'spherical retina'). Points in 3D are represented in spherical polar rather than Cartesian co-ordinates – spherical polar co-ordinates (r, θ, φ) (see Figure 10.1) are related to (x, y, z) by the equations

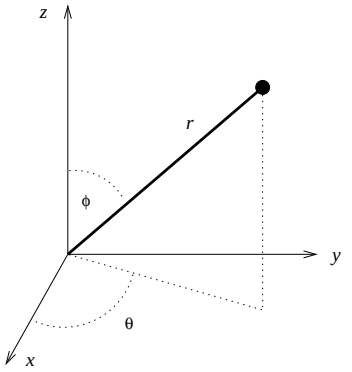


Figure 10.1: *Definition of spherical polar co-ordinates.*

$$r^2 = x^2 + y^2 + z^2 \tag{10.9}$$
$$y = x \tan\theta \tag{10.10}$$
$$z = r\cos\varphi \tag{10.11}$$

Since the image is spherical, we can specify co-ordinates as (θ, φ) pairs rather than (x, y) as usual, and the optical flow is then $(\frac{d\theta}{dt}, \frac{d\varphi}{dt})$. Supposing the observer's speed to be v (in the direction of the z axis), the motion of points in 3D is given by

$$\frac{dx}{dt} = 0 \ , \quad \frac{dy}{dt} = 0 \ , \quad \frac{dz}{dt} = -v \tag{10.12}$$

Differentiating equation (10.9) with respect to t gives

$$\begin{aligned} 2r\frac{dr}{dt} &= 2x\frac{dx}{dt} + 2y\frac{dy}{dt} + 2z\ \frac{dz}{dt} \\ &= -2vz \\ \frac{dr}{dt} &= -\frac{vz}{r} \\ &= -v\cos\varphi \end{aligned} \tag{10.13}$$

Differentiating equation (10.10) with respect to t gives

$$\begin{aligned} \frac{dy}{dt} &= \tan\theta \frac{dx}{dt} + x \sec^2\theta \frac{d\theta}{dt} \\ 0 &= 0 + x \sec^2\theta \frac{d\theta}{dt} \end{aligned} \tag{10.14}$$

and hence

$$\frac{d\theta}{dt} = 0 \tag{10.15}$$

Differentiating equation (10.11) with respect to t gives

$$\frac{dz}{dt} = \cos\varphi \frac{dr}{dt} - r \sin\varphi \frac{d\varphi}{dt} \tag{10.16}$$

and hence, by equations (10.12) and (10.13)

$$-v = -v\cos^2\varphi - r\sin\varphi \frac{d\varphi}{dt} \tag{10.17}$$

and so

$$\frac{d\varphi}{dt} = \frac{v(1-\cos^2\varphi)}{r\sin\varphi} = \frac{v\sin\varphi}{r} \tag{10.18}$$

Equations (10.15) and (10.18) are important. The former says that, for this particular motion, the rate of change of θ is zero (θ is constant). More interestingly, the latter says that given the optical flow $\frac{d\varphi}{dt}$, then the distance r of a 3D point from the observer can be recaptured up to a scale factor v. In particular, if v is known, then r, and a complete depth map, can be deduced from the optical flow. The depth map allows a reconstruction of the 3D scene and hence characteristics of surfaces (smoothly varying areas of r) and of edges (discontinuities in r) will be available.

In the case that v is not known, it turns out that surface information is still available directly from the flow. In particular, suppose a point P lies on a smooth surface, which at P may be specified by the direction of a normal vector $\mathbf{n}$. Such a direction may be specified by two angles α and β where α is the angle between $\mathbf{n}$ and a plane Π_1 defined by P and the z axis, and β is the angle between $\mathbf{n}$ and a plane Π_2 which passes through P and the origin, and is perpendicular to Π_1. Intuitively, it is clear that the rate of change of r with respect to θ and φ provides information about the direction of $\mathbf{n}$. Moderately straightforward co-ordinate geometry gives the relations

$$\tan\alpha = \frac{1}{r}\frac{\partial r}{\partial\varphi} \qquad \tan\beta = \frac{1}{r}\frac{\partial r}{\partial\theta} \tag{10.19}$$

These equations depend upon a knowledge of r (the depth map), but it is possible to combine them with equation (10.18) to overcome this. For convenience, write $\frac{d\varphi}{dt} = \dot{\varphi}$; then, by equation (10.18)

$$r = \frac{v\sin\varphi}{\dot{\varphi}} \tag{10.20}$$

and so

$$\begin{aligned} \frac{\partial r}{\partial \varphi} &= v\frac{\dot{\varphi}\cos\varphi - \sin\varphi\frac{\partial\dot{\varphi}}{\partial\varphi}}{\dot{\varphi}^2} \\ \frac{\partial r}{\partial \theta} &= -v\frac{\sin\varphi\frac{\partial\dot{\varphi}}{\partial\theta}}{\dot{\varphi}^2} \end{aligned} \tag{10.21}$$

Substituting (10.20) and (10.21) into equations (10.19) gives

$$\begin{aligned} \tan\alpha &= \cot\varphi - \frac{1}{\dot{\varphi}}\frac{\partial\dot{\varphi}}{\partial\varphi} \\ \tan\beta &= \frac{1}{\dot{\varphi}}\frac{\partial\dot{\varphi}}{\partial\theta} \end{aligned} \tag{10.22}$$

Thus, given the flow $\dot{\varphi}$ (which we assume), the angles α and β are immediately available, irrespective of S and without any need to determine the depth map given by r.

The original reference [Clocksin 80] provides full information on this derivation, and proceeds to describe how edge information may also be extracted from knowledge of the flow. It also includes some interesting discussion of psycho-physical considerations of human motion perception in the context of a computational theory.

10.1.2 Shape from texture

A further property of which there is clear psycho-physical evidence of human use to extract depth is texture [Marr 82]. To appreciate this, it is only necessary to consider a regularly patterned object viewed in 3D – two effects would be apparent; the angle at which the surface is seen would cause a (perspective) distortion of the **texture primitive** (**texel**), and the relative size of the primitives would vary according to distance from the observer. Simple examples, shown in Figure 10.2 are sufficient to illustrate this. Much use can be made of texture in computer vision at various levels of abstraction, and Chapter 14 examines them in some detail. Here we look briefly at the use of textural properties to assist in the extraction of shape [Bajcsy and Lieberman 76, Kanatani and Chou 89].

Considering a textured surface, patterned with identical texels which have been recovered by lower level processing, note that with respect to a viewer it has three properties at any point projected onto a retinal image; distance from the observer, **slant**, the angle at which the surface is sloping away from the viewer (the angle between the surface normal and the line of sight), and **tilt**, the direction in which the slant takes place. Attempts to recapture some of this information is based on the **texture gradient** – that is, the direction of maximum rate of change of the perceived size of the texels, and a scalar measurement of this rate. One approach [Bajcsy and Lieberman 76] assumes a uniform texel size was assumed; the texture gradient was

If the texture is particularly simple, the shape of the perceived texels will reveal surface orientation information. For example, if a plane is marked with identical circles they will be seen in an image as ellipses (see Figure 10.3). The eccentricity of the ellipses provides information about the slant, while the orientation of the ellipse axes indicates the tilt [Stevens 79]. There is evidence to suggest [Stevens 79] that the human viewer uses the texture gradient

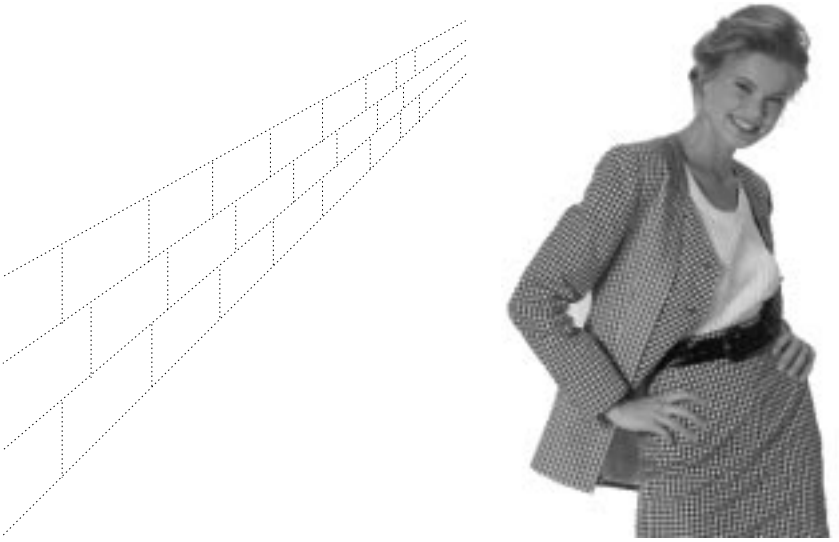


Figure 10.2: *A simple texture pattern in 3D. The left side shows a vanishing brick wall and the right the shape of a woman's body perceived from texture changes.*

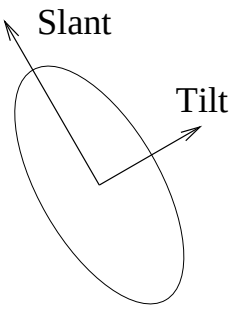


Figure 10.3: *Slant and tilt are revealed by texel properties.*

as a primary clue in the extraction of tilt and (relative) distance, but that slant is inferred by processing based on estimates of the other two parameters. Tilt is indicated by the direction of the texture gradient (see Figure 10.4), while the apparent size of objects decreases as the reciprocal of their distance from the viewer.

Large scale texture effects can provide information about large scale scene geometry; in particular, strong linear effects will indicate 'vanishing points' that may be joined to give the scene horizon. Note that it is not necessary in fact for the image to contain large numbers of long straight lines for this conclusion to be available since often such lines can be inferred by a number of segments whose co-linearity may be deduced by, for example, a Hough transform. Such co-linearity may well be a property of urban scenes in which rectangular objects, some large, can predominate.

Texture has many interpretations and there is a correspondingly large number of attempts to exploit it in shape extraction – a useful grounding may be found in [Witkin 81]. A multiple scale approach was used in [Blostein and Ahuja 89], while [Aloimonos and Swain 85] gives

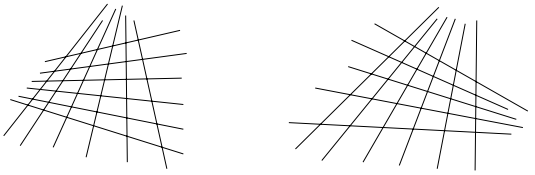


Figure 10.4: *Tilt affects the appearance of texture.*

an interesting approach in which 'Shape from texture' is shown to be equivalent to 'Shape from shading', thereby allowing the use of established results in surface parameter extraction from shading. Texture is usually used as an additional or complementary feature, augmenting another, stronger clue in shape extraction.

10.1.3 Other Shape from X techniques

Shape from focus/defocus techniques are based on the fact that lenses have finite depth of field, and only objects at the correct distance are in focus; others are blurred in proportion to their distance. Two main approaches can be distinguished: shape from focus and shape from defocus.

Shape from focus measures depth in one location in an active manner; this technique is used in 3D measuring machines in mechanical engineering. The object to be measured is fixed on a motorized table that moves along x, y, z axes. A small portion of the surface is observed by a camera through a microscopic lens, and if the surface patch in view (given by a small image window) is in focus then the image has he maximal number of high frequencies; this qualitative information about focus serves as a feedback to the z-axis servomotor. The image is put into focus and x, y, z co-ordinates read from the motorized table. If the depth of all points in the image are to be measured, a large number of images is captured by displacing the sensor in small increments from the scene, and the image of maximum focus is detected for each image point [Krotkov 87, Nayar and Nakagawa 94].

Shape from defocus typically estimates depth using two input images captured at different depth. The relative depth of the whole scene can be reconstructed from image blur. The image is modeled as a convolution of the image with a proper point spread function (see Section 2.1.2); The function is either known from capturing setup parameters or estimated, for example by observing a sharp depth step in the image. The depth reconstruction, which is an ill posed problem [Pentland 87], is performed by local frequency analysis. Depth from defocus shares an inherent problem with shape from stereo and shape from motion, in that it requires the scene to be covered by a fine texture. A real-time (30 Hz) depth from defocus sensor has been built [Nayar et al. 96]: The device uses active illumination by texture and analyzes relative blur in two images captured at two different depths. The derivation of the illumination pattern and depth estimation is posed as an optimization problem in the Fourier domain. **Shape from vergence** uses two cameras fixed on a common rod. Using two servomechanisms, the cameras can change the direction of their optical axes (verge) in the plane containing a line segment joining their optical centers. Such devices are called **stereo heads**, see Figure 10.5.

The aim of shape from vergence is to to ease the correspondence problem for estimating

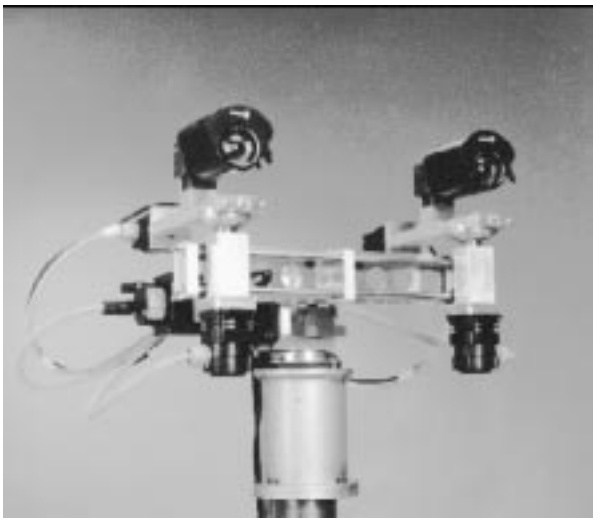


Figure 10.5: *An example of a stereo head. Courtesy J. Kittler, University of Surrey, U.K.*

depth [Krotkov and Bajcsy 93]; vergence is used to align individual feature points in both left and right images.

Shape from contour aims to describe a 3D shape from contours seen from one or more view directions. Objects with smooth bounding surfaces are quite difficult to analyze.

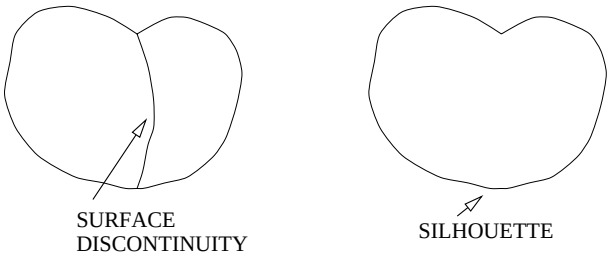


Figure 10.6: *Apricot; contour as silhouette (left) or as silhouette plus surface discontinuities (right).*

Following terminology given in [Ullman 96], assume the object is observed from some view point. The set of all points on the object surface where surface normal is perpendicular to the observer's visual ray is called a **rim** [Koenderink 90]. Note that in general the rim is not a planar curve. Assuming orthographic projection the rim points generate a **silhouette** of an object in the image. Silhouettes can be easily and reliably captured if back light illumination is used, although there is possible complication in the special case in which two distinct rim points project to a single image point.

The most general approach considers contours as silhouettes plus images of the salient curves on the surface, e.g. those corresponding to surface curvature discontinuities. The latter are often found using an edge detector from an intensity image. The trouble is that this process is not often robust enough; the simpler – and more often used – approach explores silhouettes as contours. In Figure 10.6, silhouette and surface discontinuity are shown on an

image of an apricot.

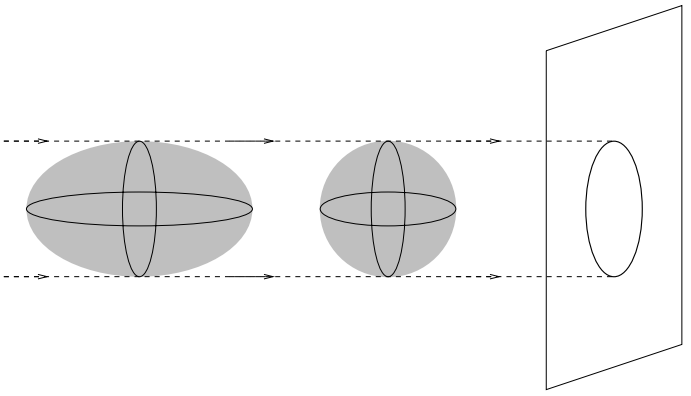


Figure 10.7: *Ambiguity of the shape from contour task.*

The inherent difficulty of the shape from contour comes from the loss of information in projecting 3D to 2D. We know that this projection is not invertible, since one image may result from the projection of different objects. This fact is illustrated in Figure 10.7, in which both a sphere and an ellipsoid project to the same image, the ellipse. Humans are surprisingly successful at perceiving clear 3D shapes from contours, and it seems that tremendous background knowledge is used to assist. Understanding this human ability is one of the major challenges for computer vision. Contours are used as constraints on the shape they represent, and the aim is to reduce the number of possible interpretations.

This section has merely formulated the shape from contour task. The reader interested in solutions should consult [Nevatia et al. 94].

10.2 Full 3D objects

10.2.1 3D objects, models and related issues

The notion of a **3D object** allows us to consider a 3D volume as a part of the entire 3D world. This volume has a particular interpretation (semantics, purpose) for the task in hand. Such an approach accords with the way general systems theory [Klir 91] treats complex phenomena, in which objects are separated from uninteresting background. Thus far, we have treated geometric (section 9.2) and radiometric (section 9.3) techniques that provide intermediate 3D cues, and it was implicitly assumed that such cues help to understand the nature of a 3D object.

Shape is another informal concept that humans typically connect with a 3D object. Recall the understanding we have when thinking of the shape of a mountain, or a vase or a cup. Computer vision aims at scientific methods for 3D object description, but there are no mathematical tools yet available to express shape in its general sense. The reader interested in abstract aspects of shape might texts on the shape of solid objects [Koenderink 90]. Here however, we shall not consider 3D shape issues in their full abstract sense. Instead, the simple geometrical approach treating parts of 3D objects locally as simple volumetric or

surface primitives is used. Curvilinear surfaces with no restriction on surface shape are called **free form surfaces**.

Roughly speaking, the 3D vision task distinguishes two classes of approach:

1. *Reconstruction* of the 3D object model or representation from real world measurements with the aim of estimating a continuous function representing the surface.

2. *Recognition* of an instance of a 3D object in the scene. It is assumed that object classes are known in advance, and that they are represented by a suitable 3D model.

The reconstruction and recognition tasks use different representations of 3D objects. Recognition may use approaches that distinguish well between distinct classes, but do not characterize an object as a whole.

Humans meet and recognize often **deformable objects** that change their shape [Terzopoulos et al. 88, Terzopoulos and Fleischer 88], an advanced topic that it too large to consider in this book.

Computer vision as well as computer graphics use **3D models** to encapsulate the shape of an 3D object. 3D models serve in computer graphics to generate detailed surface descriptions used to render realistic 2D images. In computer vision, the model is used either for reconstruction (copying, displaying an object from different viewpoint, modifying an object slightly during animation) or for recognition purposes where features are used that distinguish objects from different classes. There are two main classes of models: volumetric and surface. **Volumetric models** explicitly represent the 'inside' of a 3D object, while **surface models** use only object surfaces, as most vision-based measuring techniques can only see the surface of a non-transparent solid.

3D models make a transition towards an *object-centered* co-ordinate system, allowing object descriptions to be viewer independent. This is the most difficult phase within Marr's paradigm. Successful implementation is remote, especially compared to the success seen with the derivation of the primal and 2.5D sketches. Unlike earlier stages, there is little physiological guidance that can be used to design algorithms since this level of human vision is not well understood. Marr observes that the target co-ordinate system(s) should be modular in the sense that each 'object' should be treated differently, rather than employing one global co-ordinate system – this prevents having to consider the orientation of model components with respect to the whole. Representations based on an object's 'natural' axes, derived from symmetries, or the orientation of stick features, are likely to be of greater use.

3D models of objects are common in other areas besides computer vision, notably Computer Aided Design (CAD) and computer graphics where image synthesis is required – that is, an exact (2D) pictorial representation of some modeled 3D object. Use of an object representation which matches the representation generated by CAD systems has been an active research area for years, with substantial promise for industrial model-based vision. Progress in this area is presented in [Bowyer 92] with papers devoted to CAD-based models applied to pose estimation [Kriegman 92, Ponce et al. 92, Seales and Dyer 92], 3D specular object recognition [Sato et al. 92], and invariant feature extraction from range data [Flynn and Jain 92].

Various representation schemes exist, with different properties. A representation is called **complete** if two different objects cannot correspond to the same model, so a particular

model is unambiguous. A representation is called **unique** if an object cannot correspond to two different models. Most 3D representation methods sacrifice either the completeness or uniqueness property. Commercial CAD systems frequently sacrifice uniqueness; different design methodologies may produce the same object. Some solid modelers maintain multiple representations of objects in order to offer flexibility in design.

Due to self-occlusion of objects and to triangulation based measuring methods, most vision-based measuring sensors inherently produce only partial 3D descriptions of objects. A fusion of several such measurements from different viewpoints is needed to obtain the shape of an object entirely. An ideal 3D sensor would provide a set of 3D uniformly sampled points on the surface together with their relation to neighboring points.

10.2.2 Line labeling

Early attempts to develop 3D vision systems tried to reconstruct a full 3D representation from a single, fully segmented view of a scene. The step between the dimensions was made by assuming that all objects in the scene had planar faces (see Figure 10.8), and that three faces met at each vertex. A perfect segmentation then provides straight edged regions, and in general three of these will meet at a vertex. The idea was that this constraint was sufficient to permit a single 2D view to permit unambiguous reconstruction of a polyhedron. For obvious reasons, this is sometimes called a **blocks world** approach.

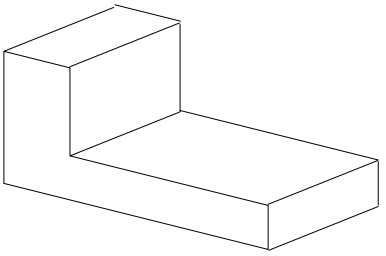


Figure 10.8: *An example blocks world object.*

The approach is clearly unrealistic for two reasons: Firstly the requirement for a perfect segmentation is unlikely to be met except in the most contrived situations; it is assumed that all edges are found, they are all linked into complete straight boundaries, and that spurious evidence is filtered out. Secondly, there is a very limited number of circumstances in which objects do consist strictly of planar faces. It is perhaps possible that industrial applications exist where both conditions might be met by constraining the objects, and providing lighting of a quality that permits full segmentation.

The idea was pioneered some time ago by Roberts [Roberts 65], who made significant progress, especially considering the time at which the work was done. Independently, two other researchers built on these ideas to develop what is now a very well known **line labeling** algorithm [Clowes 71, Huffman 71]. Mindful of the limitations of the blocks world approach, research into 3D vision has left these ideas behind and they are now largely of historical interest only. What follows is only an overview of how line labeling works, but it is instructive as firstly it illustrates how the 3D reconstruction task may be naively approached, and secondly it is good example of **constraint propagation** (see Chapter 8) in action. The algorithm

rests on observing that, since each 3D vertex is a meeting of exactly three planar faces, there are only four types of junction that may appear in any 2D scene (see Figure 10.9). In the 3D world, an edge may be concave or convex, and in its 2D projection, the three faces meeting at a vertex may be visible or occluded. These finite possibilities permit an exhaustive listing of

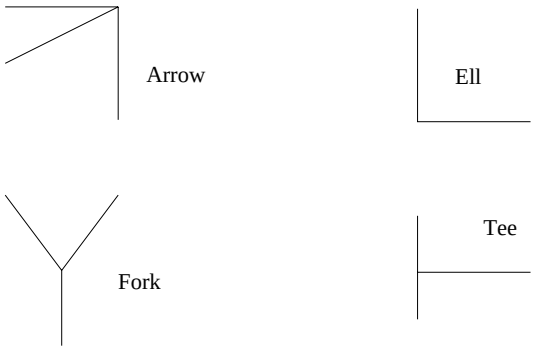


Figure 10.9: *The four possible 2D junctions.*

interpretations of a 2D vertex as a 3D vertex – there are in fact twenty-two of them [Clowes 71].

The problem now reduces to deriving a mutually consistent set of vertex labels; this may be done by employing constraints such as an edge interpretation (convex or concave) being the same at both ends, and that circumnavigating a region provides a coherent 3D surface interpretation. At a high level, the algorithm is;

Algorithm 10.1: Line labeling

1. Extract a complete and accurate segmentation of the 2D scene projection into polygons.
2. Determine the set of possible 3D interpretations for each 2D vertex from a pre-computed exhaustive list.
3. Determine 'edge-wise' coherent labelings of vertices by enforcing either concave or convex interpretations to each end of an edge.
4. Deduce an overall interpretation by requiring a circumnavigation of a region to have a coherent 3D interpretation.

Line labeling is able to detect as 'impossible' objects such as that shown in Figure 10.10a, since it would not pass the final stage of the informally described algorithm; it would not, however, register Figure 10.10b, which defies a 3D interpretation along its upper front horizontal edge, as impossible. It is also unable, in the simple form described, to cope with 'accidental' junctions which are the meeting of four or more lines (caused by chance occlusion), although these could be analyzed as special cases.

This simple approach received a lot of attention, and was extended to consider solids whose vertices may have more than three faces meeting at a vertex (such as square-based pyramids),

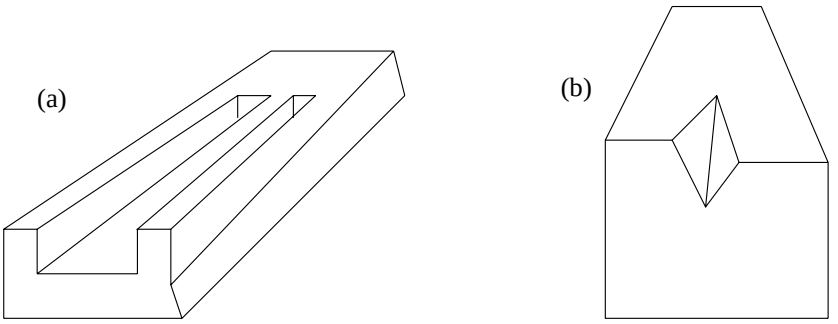


Figure 10.10: *Impossible blocks world figures.*

and scenes in which regions might represent shadows of solids [Waltz 75]. Interestingly, while the number of possible junction interpretations increases enormously, the constraint satisfaction that is required for admissible candidate interpretations prevents the resulting algorithms becoming unworkable. In general, however, line labeling exists as an interesting historical idea – the way perhaps that one might approach the problem of 3D vision as a first attempt. It is clear though that its results are limited, and fraught with problems in overcoming 'special cases'.

More recent attempts to line label interpretations may be found in [Sugihara 86, Malik and Maydan 89, Shomar and Young 94].

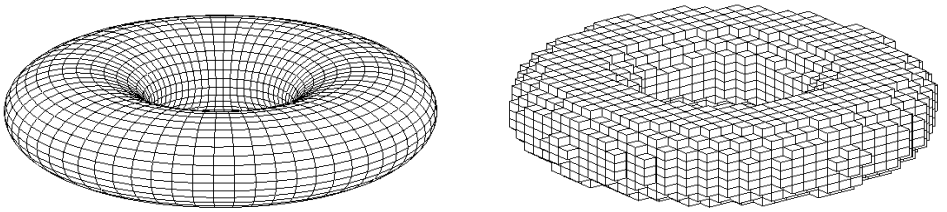
10.2.3 Volumetric representation, direct measurements

An object is placed in some reference co-ordinate system and its volume is subdivided into small volume elements called **voxels** - it is usual for these to be cubes. The most straightforward representation of voxel-based volumetric models is the **3D occupancy grid** which is implemented as a 3D Boolean array. Each voxel is indexed by its x, y, z co-ordinates; if the object is present in a particular space location the voxel has value 1 and otherwise 0. Creating such a voxel-based model is an instance of discretization with similar rules to those for 2D images; for example, the Shannon sampling theorem (see Section 2.2.1) applies. An example of a voxelized toroid is shown in Figure 10.11. One way of obtaining a voxel-based volumetric model is to synthesize it using a geometric modeler, i.e. a computer graphics program. This would permit composite objects to be assembled from some number of basic solids such as cubes, cylinders and spheres.

Another possibility is the case in which a volumetric model is to be created from an existing real object. A simple measuring technique has been used in mechanical engineering for a long time. The object is fixed to a **measuring machine**, and an absolute co-ordinate system is attached to it. Points on the object surface are touched by a **measuring needle** which provides 3D co-ordinates, see Figure 10.12. The precision depends on the machine and size of the objects, typically being $\pm$ 5 micrometers.

In simpler machines, navigation of the needle on the surface is performed by a human operator; x, y, z co-ordinates are recorded automatically. Such a surface representation can easily be converted it into volumetric representation.

Besides precision testing in machining or other mechanical engineering applications, this



(a) (b)

Figure 10.11: *Voxelization (discretization) in 3D; (a) continuous surface; (b) Voxelized image consisting of cubes of the same size.*

measuring technology may be used if the object is first created from clay by a designer. If computer aided design (CAD) is to be brought into play the 3D co-ordinates of the object are needed. An example is the automotive industry where a clay model of a car body may be created at the scale 1:1. Actually only one half of the model is produced as the car body is largely symmetric along the elongated axis. Such a model is measured on the 3D point measuring machine; as there are very many points to be measured the probe navigates semi-automatically. The points are organized into strips that cover the whole surface, and the probe has a proximity sensor that automatically stops on the surface or near to it. The probe is either a needle equipped with a force sensor or a laser probe performing the same measurement but stopping at a fixed and precise distance from the surface, e.g. 3 millimeters.

Another 3D measurement technique, **computed tomography** looks inside the object and thus yields more detailed information than the binary occupancy grid discussed so far. Tomography yields a mass density in a 2D planar slice of the object. If 3D volumetric information is required such slices are stacked on top of one another. The resulting 3D sample space consists of voxels, the values of which are mass densities addressed by the x, y, x co-ordinates. Computed tomography is used widely in medical imaging (see Chapter 16).

10.2.4 Volumetric modeling strategies

Constructive Solid Geometry (CSG)

The principal idea of CSG, which has found some success (notably with IBM's WINSOM [Quarendon 84]) is to construct 3D bodies from a selection of solid primitives. Popularly, these primitives are a cuboid, a cylinder, a sphere, a cone and a 'half space' – the cylinder and cone are considered to be infinite. They are scaled, positioned and combined by union, intersection and difference; thus a finite cone is formed by intersecting an infinite cone with an appropriately positioned half space. A CSG model is stored as a tree, with leaf nodes representing the primitive solid, and edges enforcing precedence among the set theoretical operations. The versatility of such a simply stated scheme is surprising. CSG models define properties such as object volume unambiguously, but suffer the drawback of being non-unique. For example, the solid illustrated in Figure 10.13 - a mug – may be formed by the union of the cylinder with a hole and a handle. The cylinder with the hole is obtained from a full cylinder by subtracting (in the set sense) a smaller cylinder. Further, it is not easy to model 'natural' shapes (a

Figure 10.12: *An example of a fully motorized 3D measuring machine. Manufacturer, Mitutoyo Inc., Japan.*

human head, for instance) with CSG. A more serious drawback is that it is not straightforward to recover surfaces given a CSG description; such a procedure is computationally very expensive.

Superquadrics

Superquadrics are geometric bodies that can be understood as a generalization of basic quadric solids. They were introduced in computer graphics [Barr 81]. Superellipsoids are instances of superquadrics used in computer vision.

The implicit equation for a superellipsoid is

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon_{vert}}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon_{vert}}}\right)^{\frac{\varepsilon_{hori}}{\varepsilon_{vert}}} + \left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon_{vert}}} = 1 \tag{10.23}$$

where a_1, a_2, and a_3 define the superquadric size in x, y, and z directions respectively. ε_{vert} is the squareness parameter in the latitude plane and ε_{hori} is the squareness parameter in the longitude plane. The squareness values used in respective planes are 0 (i.e. square) $\leq \varepsilon \leq 2$ (i.e. deltoid) as only those are convex bodies. If squareness parameters are greater than 2 the body changes to a cross-like shape. Figure 10.14 illustrates how squareness parameters influence superellipsoid shape.

Superquadric fitting to range images is described in [Solina and Bajcsy 90, Leonardis et al. 97], and the construction of full 3D model range images taken from several views using superquadrics is shown in [Jaklič 97]. Superquadric volumetric primitives can be deformed

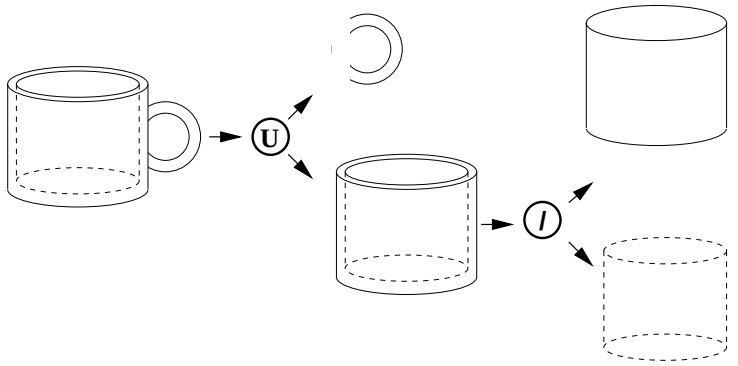


Figure 10.13: *CSG representation of a 3D object – a mug.*

by bending, twisting and tapering, and Boolean combinations of simple entities can be used to represent more complicated shapes [Terzopoulos and Metaxas 91].

Generalized cylinders

Generalized cylinders, or **generalized cones**, are often also called **sweep representations**. Recall that a cylinder may be defined as the surface swept out by a circle whose center is traveling along a straight line (spine) normal to the circle's plane. We can generalize this idea in a number of ways – we may permit any closed curve to be 'pulled along' any line in three space. We may even permit the closed curve to adjust as it travels in accordance with some function, so a cone is defined by a circle whose radius changes linearly with distance traveled, moving along a straight line. Further, the closed curve section need not contain the spine. Usually it is assumed that the curve is perpendicular to the spine curve at each point. In some cases, this constraint is released. Figure 10.15 illustrates two simple generalized cylinders.

These generalized cones turn out to be very good at representing some classes of solid body [Binford 71, Soroka and Bajcsy 78]. The advantage of symmetrical volumetric primitives, such as generalized cylinders and superquadrics, is their ability to capture common symmetries and represent certain shapes with few parameters. They are, however, ill-suited for modeling many natural objects that do not have the set of regularities incorporated into the primitives. A well known vision system called ACRONYM [Brooks et al. 79] used generalized cones as its modeling scheme.

There is a modification of the sweep representation called a **skeleton representation**, which stores only the spines of the objects [Besl and Jain 85].

10.2.5 Surface modeling strategies

A solid object can be represented by surfaces bounding it; such a description can vary from simple triangular patches to visually appealing structures such as non-uniform rational B-splines (NURBS) popular in geometric modeling. Computer vision solves two main problems with surfaces; firstly, reconstruction creates surface description from sparse depth measurements that are typically corrupted by outliers; secondly, segmentation aims to classify surface or surface patches into surface types.

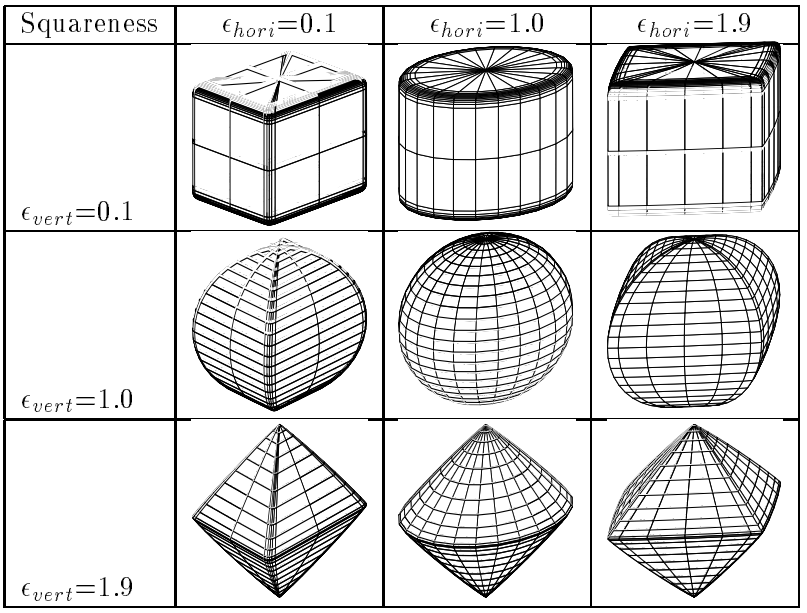


Figure 10.14: *Superellipses.*

boundary representations (B-reps) can be viewed conceptually as a triple:

- A set of surfaces of the object.
- A set of space curves representing intersections between the surfaces.
- A graph describing the surface connectivity.

B-reps are an appealing and intuitively natural way of representing 3D bodies in that they consist of an explicit list of the bodies' faces. In the simplest case, 'faces' are taken to be planar, so bodies are always **polyhedral**, and we are dealing the whole time with piecewise planar surfaces. A useful side-effect of this scheme is that properties such as surface area and

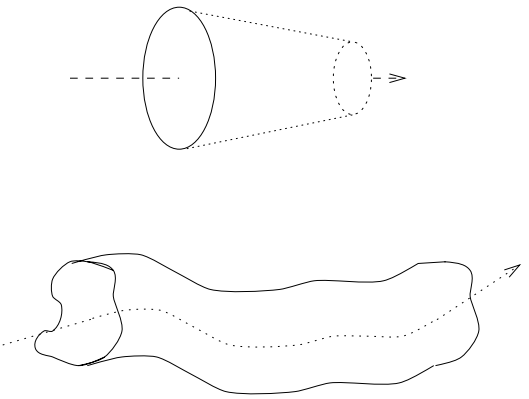


Figure 10.15: *Solids represented as generalized cylinders.*

solid volume are well defined. The simplest B-rep scheme would model everything with the simplest possible 2D polygon, the triangle. By taking small enough primitives quite satisfactory representations of complex objects can be achieved, and it is an obvious generalization to consider polygons with more edges than three.

Triangulation of irregular data points (e.g. a 3D point cloud obtained from a range scanner) is an example of an interpolation method. The best known technique is called **Delaunay triangulation**, which can be defined in two, three or more space dimensions. Delaunay triangulation is dual to the Voronoi diagram. We assume that the Euclidean distance between data points is known; then points that are closer to each other than to other points are connected. Let $d(P,Q)$ be the Euclidean distance between points P and Q, and S be the set of points $S=\{M_1,\ldots,M_n\}$. A Voronoi diagram on the set S is a set of convex polyhedra that covers the whole space. The polyhedron V_i consists of all points that are closer to the point M_i than to other points of S.

$$V_i = \{p;\ d(p,M_i) \leq d(p,M_j) \text{ for all } j = 1,2,\ldots,n\} \tag{10.24}$$

An algorithm to compute Delaunay triangulation can be found in [Preparata and Shamos 85]. A problem with Delaunay triangulation is that it triangulates the convex hull of the point set; constrained Delaunay triangulation [Faugeras 93] can be a solution.

We shall illustrate the idea of Delaunay triangulation on the simplest case of 2D planar point set (see Figure 10.16). The task is to find triangles that cover all data points in such a way that the circumcircle of any one triangle contains only the three points that are vertices of that particular triangle. The triangulation has the following properties:

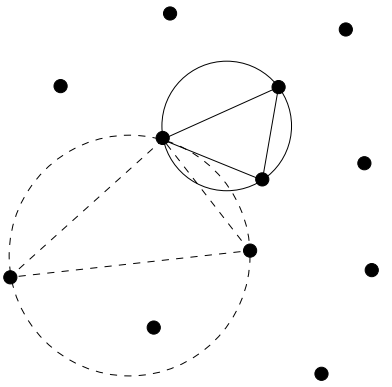


Figure 10.16: *2D Delaunay triangulation. The solid triangle belongs to the Delaunay triangulation, but the dotted one does not as its circumcircle contains an additional point.*

- The boundary of the set of points covered by triangles corresponds to the convex hull of the point set.
- The incremental algorithm constructing a triangulation of N points has expected time complexity $\mathcal{O}(N \log N)$ [Gubais et al. 92].
- The 2D Delaunay triangulation algorithm provides a unique solution if no more than three points lie on one circle.

A drawback with polyhedral or triangulated B-reps is that the concept of 'face' may not be well defined. A face should have no 'dangling' edges, and the union of a body's faces should be its boundary. Unfortunately the real world is not cooperative and many (simple) bodies exist in which face boundaries are not well defined.

The next step in generalizing descriptions of surface patches is the **quadric surface model**. Quadric surfaces are defined using second degree polynomials in three co-ordinates x, y, z. In implicit form, the equation has up to 10 coefficients and represents hyperboloids, ellipsoids, paraboloids, and cylinders.

$$\sum_{i,j,k=0\ldots2} a_{ijk}\, x^i\, y^j\, z^k = 0 \tag{10.25}$$

More complicated objects may be created from quadric surface patches. *Parametric bicubic surfaces*, defined by bivariate cubic polynomials are used in CAD systems; the commonly used Bézier surfaces fall into this category. These surfaces have the advantage that surface patches can be smoothly joined along the intersection curves, and undesirable curvature discontinuity artifacts are thus avoided. Such an approach permits much greater flexibility in the description, but it becomes important to restrict the number of possible face edges in order to limit the complexity of the computations involved.

There is an independent discipline called **geometric modeling** that considers object representation issues from the designer's point of view.

10.2.6 Registering surface patches and their fusion to get a full 3D model

A **range image** represents distance measurements from an observer to an object; it yields a partial 3D description of the surface from one view only. It may be visualized as a relief made by a sculptor – shape information from different views, e.g. from the other side of the object, is not available. Techniques for range image acquisition have been mentioned in Section 9.2.12.

Several range images are needed to capture the whole surface of an object. Each image yields a point cloud in the co-ordinates related to the range sensor, and successive images are taken in such a way that neighboring views slightly overlap, to provide information for later fusion of partial range measurements into one global, object-centered, co-ordinate system.

A fusion of partial surface descriptions into global object-centered co-ordinates implies known geometric transformations between object and sensor The process depends on the data representing one view, e.g. from simple point clouds, triangulated surfaces, to parametric models as quadric patches.

Range image registration finds a rigid geometric transformation between two range images of the same object captured from two different viewpoints. The recovery can either be based on explicit knowledge of sensor positions, e.g. if it is held in a precise robot arm or on geometric features measured from the overlapped parts of range data. Typically, both sources of information are used; an initial estimate of the appropriate geometric transformation can be provided by image feature correspondence, range image sensor data, an object manipulation device or in many cases by a human operator.

This 3D model reconstruction task has been approached by several research groups in recent years, and many partial solutions have been proposed, e.g. [Hoppe et al. 92, Higuchi

et al. 95, Uray 97]. We will present here one of the possible approaches to the task. The method automates the construction of a 3D model of a 3D free form object from a set of range images as follows:

1. The object is placed on a turn table and a **set of range images from different viewpoints are measured** by a structured light (laser plane) range finder.
2. A triangulated surface is constructed over the range images.
3. Large data sets are reduced by **decimation** of triangular meshes in each view.
4. Surfaces are registered into a common object centered co-ordinate system and outliers in measurements are removed.
5. A full 3D model of the object is reconstructed by a surface fusion process.

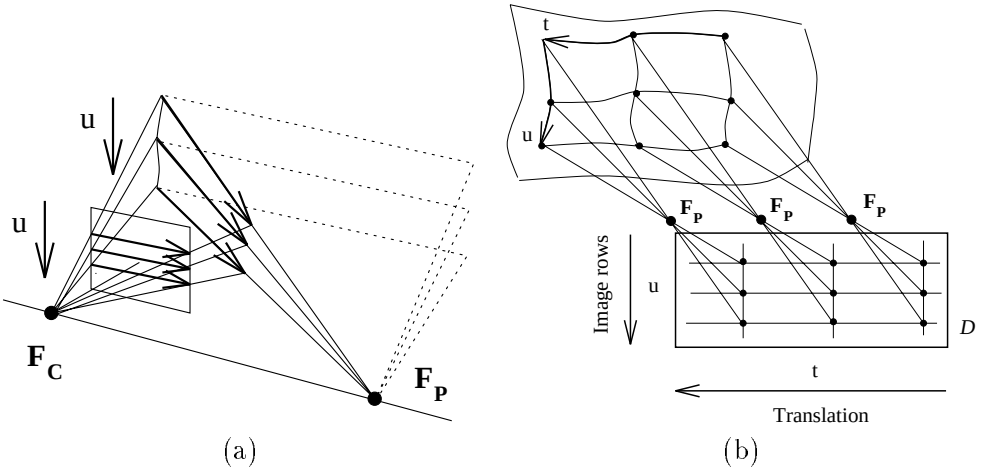


Figure 10.17: *Surface parametrization is composed from the projected laser ray and translation.*

Measurement from a laser plane range finder provides a natural connectivity relation of points along stripes. Their parameterization allows an easy construction of a 4-connected mesh following parametric curves just by connecting points found in the neighboring rows of the image and the points in neighboring scans with the same image row co-ordinate. The parameterization obtained of the measured surface is shown in Figure 10.17. The assumption of surface continuity is implemented as a restriction on the distance of neighboring points; only neighbors closer than a predefined ϵ are considered to lie on one surface next to each other. Points with no close neighbors are assumed to be outliers and are removed from data.

A 4-connected mesh cannot represent all objects, e.g. a sphere cannot be covered by a 4–sided polygons. By splitting each polygon by an edge a *triangulation* of the surface, that is able to represent any surface, is easily obtained. There are two ways to split each polygon; it is better to choose the shortest edge because then triangles with larger inner angles are obtained.

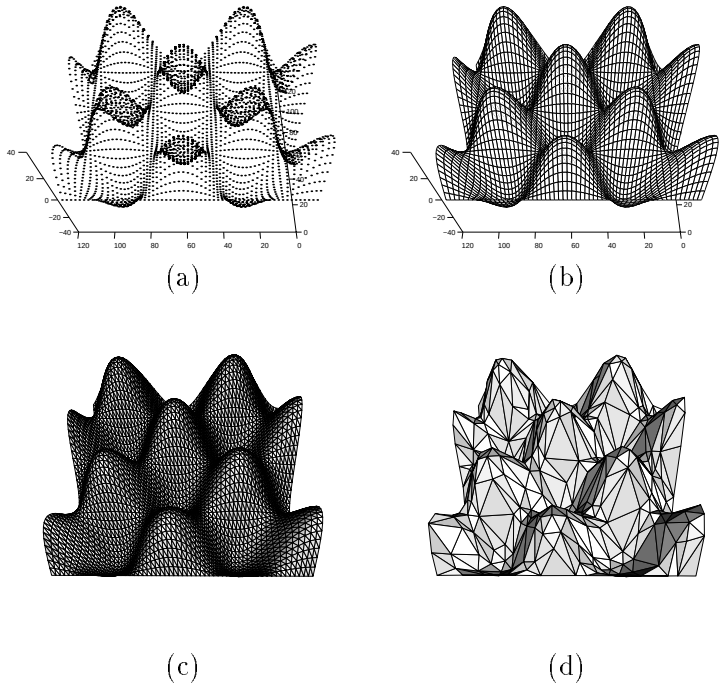


Figure 10.18: *Surface construction and its decimation shown on a synthetic sinusoidal pattern; (a) Point cloud, (b) 4-connected mesh, (c) Triangulated surface, (d) Surface after decimation of a number of triangles. Courtesy T. Pajdla, D. Večerka. Czech Technical University, Prague.*

Often, we wish to reduce the number of triangles representing the surface in areas where curvature is low [Soucy and Laurendeau 96]. Data reduction is particularly useful for the registration of neighboring views since it has worst case complexity $O(N^2)$ in the number of points. We formulate the task as a search for the best approximation of a triangulated surface by another triangulated surface that is close to the vertices of the original mesh [Hoppe et al. 92]. For instance, we might look for the closest triangulated surface with maximally n triangles, or we might want simultaneously to minimize n and a residual error to get a consensus between the precision and space costs using the Minimum Description Length principle (MDL) [Rissanen 89]. The surface triangulation procedure and node decimation is demonstrated on the synthetic pattern in Figure 10.18. Decimation of a triangulated surface from a real range image is shown in Figure 10.19.

Integration of partial shape descriptions attempts to get known geometric transformations among the views in order to register them and express them in a common co-ordinate system. Precise alignment of the data can be done automatically by gradient minimization provided that good starting transformations are available. In some cases, matching based on invariant features detected on the visual surfaces can be used [Pajdla and Van Gool 95], however, no method able to cope with a high variety of surfaces has yet been developed.

Figure 10.20 shows approximate manual surface registration, with the help of an user

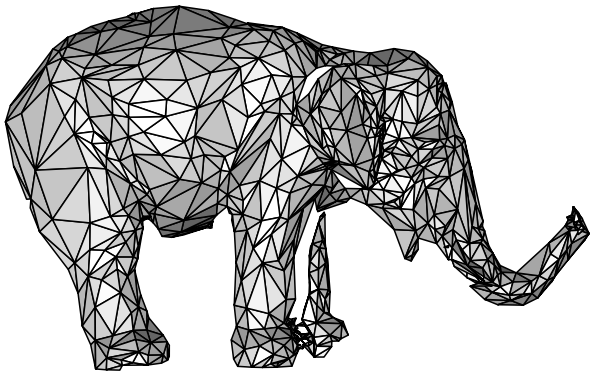


Figure 10.19: *Triangular mesh decimation for one range view of an real object – a small ceramic sculpture of an elephant. Courtesy T. Pajdla, P. Krsek. Czech Technical University, Prague.*

interaction. The mutual position of two surfaces is defined by aligning three pairs of matching points; the user selects a few point pairs (the minimum is three) on the surfaces. The approximate registration is obtained by moving one of the surfaces so that the sum of squared distances between the matching point pairs is minimal. An interactive program Geomview (authored by the Geometry Center, University of Minesota) for 3D surface viewing and manipulation was used to let the user do the registration.

When two partially overlapping surface patches are roughly registered, automatic refinement of the registration follows. It is assumed that two partially overlapping surfaces P and X related by a global transformation are available. In our case, all transformations are subgroups of a projective group in $\mathcal{P}^3$. Surface registration looks for the best Euclidean transformation T that overlays P and X. T is found by minimization of

$$e = \min_T \rho(P, T(X)) \tag{10.26}$$

where ρ is a cost function evaluating the quality of match of two surfaces. In Euclidean geometry, it might be the distance between the points on a surface.

The **iterative closest point algorithm** (ICP) developed by Besl and McKay [Besl and McKay 92] solves the registration automatically provided a good initial estimate of T is available. The algorithm assumes that one of the surfaces is a subset of the second, meaning that only one surface can contain points without correspondence to the second surface. ICP is an iterative optimization procedure that looks for the geometric transformation of one surface best to match the second. It is likely that the cost function will be non-convex, and so there is a consequent danger of falling into local minima - thus a good initial estimate is needed.

We present here a modification of the ICP algorithm which is able to register partial corresponding surfaces. This approach uses the idea of reciprocal points [Pajdla and Van Gool 95] to eliminate points without correspondence. Assume point $\mathbf{p}$ is on the surface P and that y is the closest point on the surface X. The closest point on the surface P to $\mathbf{y}$ is the point $\mathbf{r}$ (see Figure 10.22). Points $\mathbf{p}$, satisfying the condition that the distance is lower then ϵ are called ϵ-reciprocal – only these points are registered. Let P_ϵ, denote the set of ϵ-reciprocal points on the surface P; then the ICRP algorithm is:

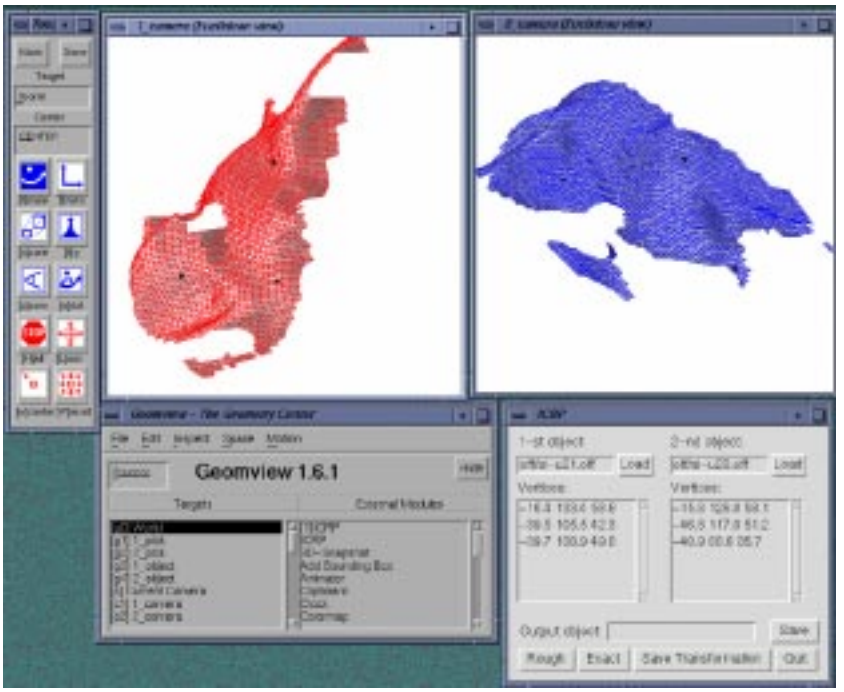


Figure 10.20: *Manual registration of surfaces in Geomview.*

Algorithm 10.2: Iterative Closest Reciprocal Points

1. Initialize $k = 0$ and $P_0 = P$.
2. Find closest points Y_k for P_k and X .
3. Find reciprocal points $P_{\epsilon 0}$ and $Y_{\epsilon k}$.
4. Compute the mean square distance d_k between $P_{\epsilon k}$ and $Y_{\epsilon k}$.
5. Compute the transformation T between $Y_{\epsilon k}$ and $P_{\epsilon 0}$ in the east squares sense.
6. Apply the transformation T: $P_{k+1} = T(P_0)$.
7. Compute the mean square distance $d_{k'}$ between $P_{\epsilon k+1}$ and $Y_{\epsilon k}$.
8. Terminate if the difference $d_k - d_{k'}$ is below a preset threshold or if the maximal number of iterations is exceeded; otherwise go to step 2.

When visual surfaces are properly registered, surface *integration* follows. All partial measurement will have been registered and can be expressed in one object-centered co-ordinate system, and constitute a global point. A problem is that the 3D object representation was created from overlapping surface patches corresponding to several views; these patches were integrated following one traversal around the object. All measurements are corrupted by

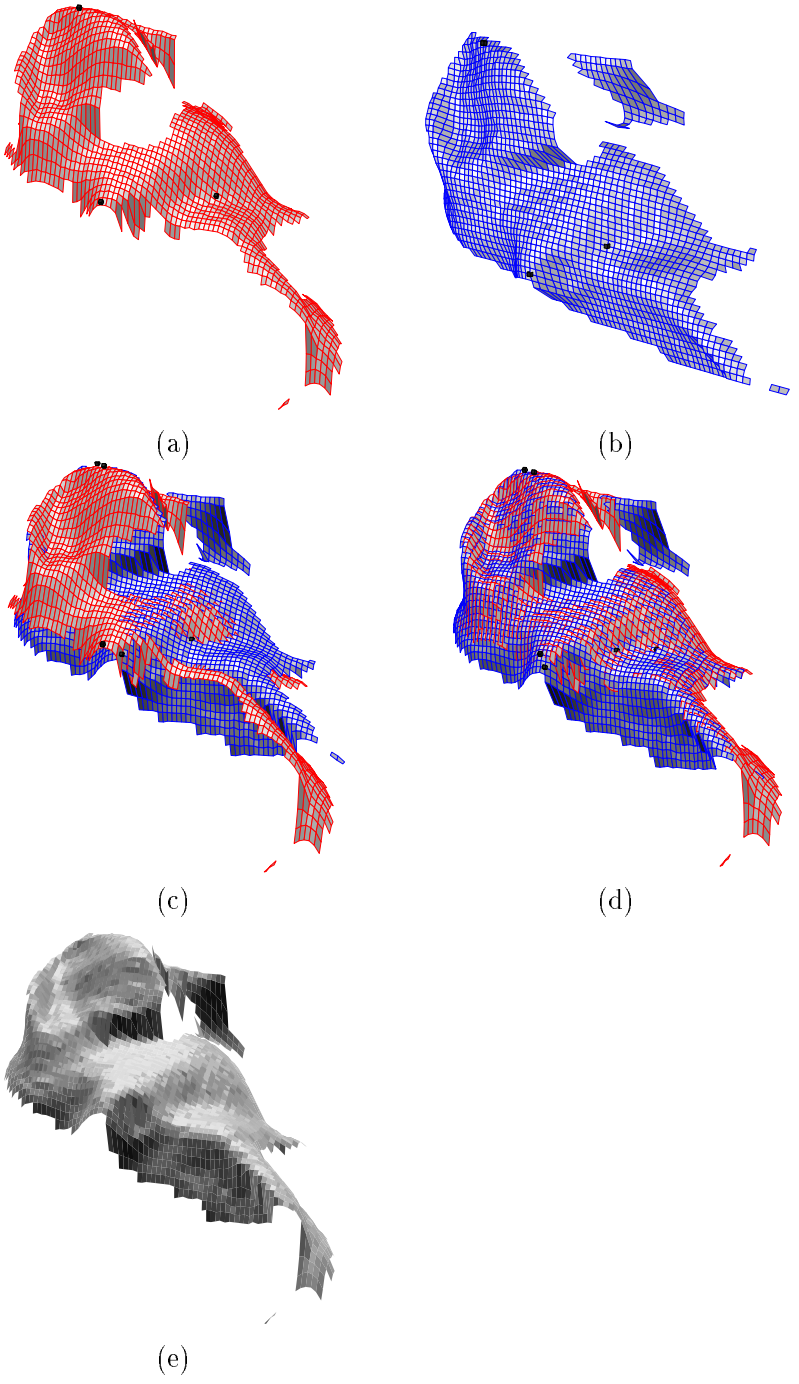


Figure 10.21: *The process of registration: (a) three points on the first surface, (b) three points on the second surface, (c) surfaces after rough registration, (d) after exact registration, (e) rendered result. Courtesy T. Pajdla, D. Večerka. Czech Technical University, Prague.*

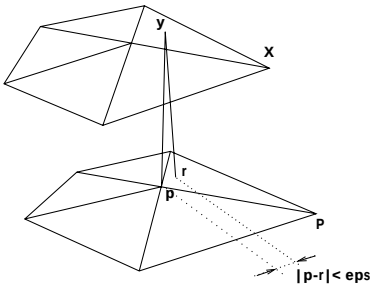


Figure 10.22: *The notion of the closest point of the surface for introducing reciprocal points into the ICP algorithm.*

some noise, and this noise is propagated from one surface patch to the other. The important issue is to have approximately the same error when joining the first and last patches. To ensure that the global error is minimal, registered surface points should be rearranged during the surface integration to maintain global consistency.

The next task is to create an analytic shape description of the object's surface by approximation, using some implicit or explicit formulae; commonly, this is not done for the whole object but just for parts of it. A recent promising [Šára and Bajcsy 98] uses a multiscale local noise model. An uncertainty ellipsoid (called a fish scale) is used to integrate local information of the point cloud on the surface into a global shape description. The fish scales can be created at multiple resolutions, and their overlap and consistency is explored when creating a 3D shape model.

10.3 3D model-based vision

10.3.1 General considerations

The recognition approach to 3D vision aims at successful recognition of real-world objects using standard pattern recognition approaches. The tremendous complexity of understanding the 3D scene which is needed for 3D object reconstruction is avoided. The key issue is foreknowledge about specific objects under inspection – the classes of objects in which we are interested are represented by **partial models**.

Object models contain more information than sensor data extracted from one view, and it is therefore not possible to transform sensor data into a complete model data representation. Fortunately, sensor data can be matched with partial model data. It is useful to work with an intermediate representation scheme computable both from sensor and model data which permits the reduction of the large amount of sensor data. A matching between an object and the model is then carried out at this intermediate representation level. The best matching occurs when the hypothetical object model configuration represents the scene world as accurately as possible. Accuracy might then be measured quantitatively by **matching errors**. This difference might be used to control the whole recognition process by closing a feedback loop.

A selection of lower level processing techniques will generate some selection of clues and features, and from some model base there will be a selection of objects which may or may not

match the observations. Most model-based recognition systems are domain oriented and do not provide a general solution. This approach is particularly useful for industrial and other applications.

A typical object recognition system operates in four phases:

- Data acquisition;
- Feature detection;
- Hypothesis generation;
- Hypothesis verification.

Earlier chapters have discussed several approaches to **matching**, such as hypothesize and verify, efficient pruning of interpretation trees, graph (subgraph) matching, properly indexed database, etc. In Section 10.2.2, the line labeling approach to the recapture of 3D was outlined, and it was explained why in general this solution is inadequate. **Verification** can be approached in two ways:

- At a data level, where synthetic data is created from a hypothesized object, and its position and orientation are compared with the actual input measurements;
- At a feature level. Some metric is used to compare features from a hypothesized and a real object.

Here we shall make some general remarks about 3D object recognition, and by way of example present in detail for didactic a well known algorithm that solves the problem for the simple case of a well-understood object with planar, straight edged faces by replacing the 3D problem by many 2D ones [Goad 86].

10.3.2 Goad's algorithm

A particular model matching algorithm due to Goad [Goad 86] has received a lot of attention. Goad's algorithm is interesting since it is simple enough in principle to show how the task may be done, but generates enough complexities in implementation to illustrate how difficult the task of model matching, even in a relatively simple case, can become.

Goad's algorithm aims to recover the 3D co-ordinates and orientation of a known polyhedral object from a single intensity image. The object is 'known' in the sense that its edges, which are assumed to be straight, and their relative position to each other, are exhaustively listed in the object model with which the algorithm is provided. This contrasts with the easier problems of locating a known object of known orientation (a common aim in industrial vision systems), and the harder one of locating imprecisely defined objects of unknown orientation (a more widely applicable, and elusive, solution).

Following the terminology of the original reference, an object edge will refer to a straight line segment in 3D that forms part of the boundary of an object face. Projections of these into 2D (the image) are referred to as lines. The algorithm proceeds on a number of assumptions:

1. It is assumed that an edge and line detector have done their work and the lines (straight boundaries) in the image have been extracted. The algorithm permits these extracted

lines to be imprecise within certain bounds, and, in its full form, is able to make allowances for spurious and missing evidence (that is, lines where there should be none and no lines where there ought to be).

2. The object to be located is either fully in the field of view, or not visible at all.

3. The distance to the object is known; this permits the further assumption that the camera lies at some point on a sphere centered at the origin of an object-based co-ordinate system. Without loss of generality, we assume this is a unit sphere.

4. The field of view is sufficiently narrow to permit the assumption that changing the **orientation** of the camera at a given position only causes the features in view to undergo a simple rotation and translation. While such a change in orientation may affect which features are visible, the lengths of lines in view will not alter, within a small tolerance.

The general strategy of the algorithm is intuitively simple – an edge of the object is taken and a likely match for it found in the image. This possible match will not constrain the position of the object completely – there will be a range (a locus) of camera positions that is consistent with what is observed. Now select another edge from the model; the restrictions provided by the (putatively) matched edge will limit the possible position of the projection of this new edge into the image; we may thus predict the position of this edge in the image, within bounds governed by the accuracy of measurements and line finders. If this projected edge cannot be located, the supposed match is false. If it can, it may be expected to restrict further the locus of possible camera positions that is consistent with all hitherto deduced possible camera positions – the **observation** is used to **back project** and thereby reduce the possible locus. Another edge is now selected and the procedure repeated iteratively until either the match fails, or there is strong enough evidence (sufficient matched edges, and a restricted enough locus) to deduce that the match is correct and thereby specify the object's location and orientation. Early in the matching process, with few object edges matched, the bounds on the prediction may be very wide; as the match proceeds so the predictions will become more precise. When the match fails, we may backtrack to a point where more than one image edge was a possible match for an object edge and try again.

This 'Predict-Observe-Back Project' cycle is a simple instance of an elementary matching algorithm – sequential matching with backtracking [Hayes-Roth et al. 83], and is a typical example of a top-down, hypothesize and verify approach.

Some notation and definitions are necessary to proceed. Remember we are working in object-centered co-ordinates, so the object is regarded as fixed while the camera position and orientation are the unknowns to be determined. Throughout, an object edge will be regarded as an **oriented** line segment, given by an ordered pair of co-ordinates, while an image line may be unoriented. Thus an object edge is an **ordered** pair of (3D) co-ordinates, while an image line is given by a (perhaps unordered) pair of (2D) co-ordinates.

Let $\mathbf{p}$ be a 3D (camera) position, and $\mathbf{q}$ a 3D (camera) orientation. $\mathbf{p}$ is just a 3D co-ordinate, which we are constraining to lie on the unit sphere. If e is an object edge, let $P([\mathbf{p},\mathbf{q}],e)$ denote the **oriented** image line which results from viewing e from $\mathbf{p}$ at orientation $\mathbf{q}$, using a perspective transformation. $P([\mathbf{p},\mathbf{q}],e)$ may be undefined if e is occluded, or the projection is outside the field of view. If it is defined, it is an **ordered** pair of 2D co-ordinates.

The possible positions on the surface of the unit sphere are quantized – a set of such positions will be referred to as a **locus**. A given edge e will only be visible from some of these positions, which is referred to as the **visibility locus** of e.

An assignment between object edges and image lines will be called a **match** M. For an object edge e, $M(e)$ will denote the oriented image line assigned to it by M; for some e, $M(e)$ may be undefined, so a match need not be a complete assignment of object edges. Then a match M is **consistent** with a camera position and orientation $[\mathbf{p}, \mathbf{q}]$ if for each object edge e we have $P([\mathbf{p}, \mathbf{q}], e) = M(e)$ to within errors of measurement. A match M is consistent with a camera position $\mathbf{p}$ if there is some orientation $\mathbf{q}$ such that M is consistent with $[\mathbf{p}, \mathbf{q}]$. A match is consistent with a locus L if it is consistent with every position of that locus. L is initialized from the assumptions deduced from the match of the first edge.

In overview, the algorithm is then;

Algorithm 10.3: Goad's matching algorithm

1. Initialize by finding a plausible match in the image for one edge.
2. For the current match M and the current locus L, select an unmatched edge e.
3. By considering a matched edge e_0, compute bounds on the possible position of $P([\mathbf{p}, \mathbf{q}], e)$ relative to $P([\mathbf{p}, \mathbf{q}], e_0)$ as $\mathbf{p}$ ranges over L (this position depends only on $\mathbf{p}$ and not on $\mathbf{q}$ from assumption 4 above). Thus determine a range of possible positions for $M(e)$ in the image.
4. If a candidate for $M(e)$ is located, back project – that is restrict the locus L to L' by rejecting all points in L that are not consistent with the measured position of $M(e)$. L' is those points $\mathbf{p}$ in L from which the predicted position of $P([\mathbf{p}, \mathbf{q}], e)$ relative to $P([\mathbf{p}, \mathbf{q}], e_o)$ is the same as the position of $M(e)$ relative to $M(e_0)$ to within measurement error.
5. If more than one candidate for $M(e)$ is located, mark this as a choice point for future backtracking.
6. If no candidate for $M(e)$ is located, backtrack to the last choice point.
7. Iterate until the match is regarded as certain.

It is acknowledged that the image line detector is not going to be perfect. Some e, although expected to be in view, will not have a match $M(e)$ in the image. Two measures are maintained as the algorithm proceeds that gauge whether its oversights are 'reasonable', and whether it is likely to be 'complete'. These are:

1. **Reliability**: The probability that the edges making up the match to date arose by chance from background information is calculated and the inverse of this probability is called the reliability of the match. When the reliability exceeds a certain threshold, the match is regarded as correct and the algorithm terminated. These probabilities may best be computed on the basis of statistics gathered from images of the same class as that under examination.

2. **Plausibility**: Assuming the match is correct, and has missed some edges, the probability that those edges would indeed have been missed by the line detector in use is calculated – these probabilities assume knowledge of the performance of the line detector which once again are best accumulated from running it on sample images.

Now high reliability indicates that the match is correct, while low plausibility indicates it is probably incorrect (although we must beware – high plausibility does not imply a correct match and low reliability does not imply it is incorrect). Plausibility is introduced into the algorithm by requiring that if it falls below a certain threshold, then we must backtrack. In fact, this generates another possible choice point – if we assume e is visible, search for it and fail to find it, we may assume it should be visible but is absent from the image and proceed with reduced plausibility accordingly. Only if this assumption leads to no match do we backtrack, and consider whether the edge should be visible at all.

Edge **visibility** considers from which points of L an edge e may actually be seen – (that is, whether the visibility locus V of e intersects L). This provides another possible choice point for backtracking; first assume e is visible (that is, restrict L to its intersection with V) and proceed as described above. If a match is not found, and we require to backtrack, at this point we can now assume e is not visible and restrict L to its intersection with the complement of V. 'Visibility' needs to be defined with caution here – an edge is only regarded as visible if it is likely to be found by the line detector. Very short lines (object edges viewed nearly 'end on' for instance) would not meet this criterion.

A feature of the problem that this algorithm is designed to solve is that the object sought is modeled precisely. This fact may be exploited to speed up what would otherwise be at best a ponderous execution by going through a 'set-up' phase during which the features of the object are coded into the algorithm to be exploited at run time. Goad refers to this as the 'compile-time' of the algorithm.

There are several ways we may exploit this compile-time;

1. From a given position $\mathbf{p}$, we require during the algorithm to determine bounds on the position of an edge e relative to a matched edge e_0. This relative position, $relpos(e, e_o, \mathbf{p})$ depends only on the object, which is fully characterized independent of the run. These relative position bounds may therefore be computed at compile-time and stored for lookup. A complete, reliable and plausible set of constraints on *relpos* is proposed in [Bray and Hlavac 91].

   In fact we require $relpos(e, e_0, \mathbf{p})$ for all $\mathbf{p} \epsilon L$; this is easily done by taking the union of the bounds for each such $\mathbf{p}$. This table can also be used for the back projection; given a likely $M(e)$ we need only determine for which $\mathbf{p} \epsilon L$ this $M(e)$ is within the bounds predicted by $relpos(e, e_0, \mathbf{p})$.

2. When selecting the next edge to match, care should be taken to maximize the return on the effort put into trying to match it. This may be ensured by maximizing the likelihood of the selected edge being visible (and findable by the line detector), and by requiring that the measurements of the image position of the observed edge should provide as much information as possible about the camera position (so the locus is reduced as much as possible by making a match).

These judgements may be made at compile-time. Supposing a uniform distribution of camera positions around the sphere (in fact, allowances can be made if this is an unreasonable assumption) then the probability of the visibility of a given edge over any locus can be pre-computed. Likewise, the 'value' of measuring the position of a given edge can be computed at compile-time. If we determine a way of combining these factors by appropriately weighting the values determined (and this is not necessarily straightforward), the 'best next edge' to match, given a particular partial match, can be determined at compile-time. Goad observes that this particular pre-computation may be very expensive.

3. The elemental contributions to the plausibility measurements can also be pre-computed.

There is no doubt that performing the compile-time operations outlined will be very expensive, but this should not worry us since the expected run-time efficiency gain will make the effort well worth the cost. It is a familiar idea to pay the price of lengthy compilation in order to generate efficient running code.

Goad's algorithm is in principle quite simple – there is a variety of things we may expect to see, and they are sought in a 2D projection. Nevertheless, when examined with all its ramifications, it should be clear that the algorithm's implementation is not quite so simple.

When running, Goad managed to get respectable efficiency from his system. To deduce a complete match, it is really only necessary to make reliable matches for four object edges. He reports that on an 'average 1 MIPS machine' (remember this work is dated 1983), one matching step will take of the order of a few milliseconds, permitting several hundred attempts at matching every second. The run-time for a complete match is quoted at approximately one second, but this excludes edge and line detection. As has been remarked, much of this efficiency has been achieved at the expense of a very long 'compile-time'.

The algorithm has actually been applied to several problems – single occurrences of objects such as a connecting rod or universal joint have been located in cluttered scenes, and, more interesting, key caps (the plastic keys from a keyboard or typewriter) have been located in an image of a pile of caps. In industrial terms this problem, often referred to as bin-picking, is unreasonably difficult – multiple occurrences of the target object at multiple orientations, many partially occluded (remember the first assumption above, that the object is visible either fully or not at all). The algorithm succeeds despite the fact that the background consists of features very similar to those composing the target, and that the target has few distinguishing features.

Goad's algorithm turns out to be quite powerful. The idea of 'pre-compiling' the object description (special purpose automatic programming) produces systems of acceptable efficiency. Various elaborations exist which are not explored here, such as exploiting recurring patterns or symmetries in the object, or variable camera-to-object distance. Remember this object location is done by two-dimensional matching; that it works despite unknown orientation and is dependent on complete and thorough knowledge of the image and line detector properties, and the target object. Various elaborations on the ideas presented here have been developed [Lowe 85, Grimson 89, Bray and Hlavac 91]

10.3.3 Features for model-based recognition of curved objects from intensity images

Consider the case in which 3D curved objects are to be recognized using 2D intensity images – this is hard as it is too ambiguous. Salient curves on the object surface are believed to be seen as identifiable curves in images. These are often silhouettes or curves corresponding to surface discontinuities as steps or crest lines. The dependence on illumination causes problems. Sometimes image curves due to shadows are more visible in the image than those due to features of interest.

The **curvature primal sketch** [Brady 84] is a systematic domain independent method permitting the description of curvilinear objects. Quantities derived from differential geometry such as lines of curvature, asymptotes, bounding contours, surface intersections and planar surface patches are used. Principal curvatures and principal directions are computed. An ad hoc breadth-first method is used to link the principal direction at each point into the line of curvature.

Typically, curvature is a primary feature, and there are several ways to compute it for a discrete curve [Worring and Smeulders 93] Good success in curvature estimation based on smoothing in different scales by Gaussian filters [Lowe 89] has been observed. Such estimates are significantly biased as the smoothed curve shrinks; this shrink can be compensated for [Hlaváč et al. 94], and the resulting algorithm is easy to use. Mokhtarian [Mokhtarian 95] introduced the notion of curvature scale-space that permits the analysis and recognition of curves at multiple resolutions.

A systematic method exists which permits the use of theorems from continuous mathematics for digitized objects The basic idea is first to approximate the discrete curve by an analytic function in an explicit form; polynomials are often used for this. The differential characteristics are then computed using analytic formulae.

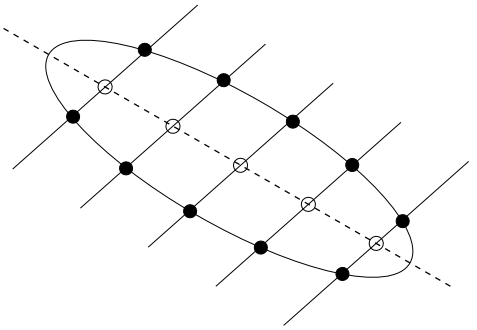


Figure 10.23: *Midpoints of parallel chords across a conic lie on a straight line*

Weiss' approximation to general curves by conic sections [Weiss 88, Weiss and Rosenfeld 96] is known to be robust; this is extendible to curved surfaces, where quadric patches are used. The approximation observes that midpoints of chords across a conic lie on a straight line (see Figure 10.23). Having a reliable approximation of the curve by conic sections, Weiss proceeds to use differential and algebraic **invariants** to perspective projection. These invariants provide features on which to base recognition of 3D models from 2D images – Weiss' invariants are based on derivatives up to the fourth order [Rivlin and Weiss 95]. The

practical disadvantage of this method is that it does not work well for curves that are close to straight lines.

A very rich description may be based on properties of object surfaces; typically these will range images Section 10.3.4. Here we present similar techniques suitable for intensity images. The term **surface features** (or characteristics) refers to descriptive features of a general smooth surface; then **surface characterization** refers to the process of partitioning a surface into regions with equivalent characteristics. A very similar process might be applied to an intensity image, and a function describing an intensity image has the same mathematical form as a surface, i.e. a function of two variables. What makes the problem difficult is the separation of illumination effects.

The symbolic scene description features should be invariant to translations and rotations of object surfaces. Differential geometry-based features are useful if they can be reliably computed from the sensor data, but this is not usually the case and surface (or curve) approximations should be adopted first.

One interesting possibility useful both for an intensity image and range images is **topographic characterization** of the surface. At each pixel in an image, a local facet-model bicubic polynomial surface is fitted to estimate the first three partial derivatives. The magnitude of the gradient, eigenvalues of a 2×2 Hessian matrix (the matrix of second derivatives) and directions of its eigenvectors are computed; these five numbers are then used to characterize the surface at the pixel. The following ten labels are used [Besl and Jain 85]; peak, pit, ridge, ravine (valley), saddle, flat (planar), slope, convex hill, concave hill, and saddle hill. This is called a **topographic primal sketch**. The main advantage of these labels is the invariance to monotonic gray level transformations such as change in brightness and contrast.

10.3.4 Model-based recognition based on range images

The recognition of simple polyhedral or piecewise quadric models has been attempted out by several researchers, e.g. [Chen and Kak 89, Newman et al. 93]; a useful review can be found in [Arman and Aggarwal 93].

We take a further step down the road of complexity consider *free form objects*. The common objects we meet everyday, such as the bust of a person, a car, a banana, etc, fall into this class. There are objects with which computer vision still cannot deal, such as flowers, bushes, or hairy piece of cloth, due to complexity and problems of range data acquisition.

The traditional approach attempts to approximate a free form surface by set of planes; for example, Faugeras and Hebert [Faugeras and Hebert 86] used planar surfaces to approximate models. Creases between planar facets and vertices were used as features for finding pairings between object and model.

The idea of the **extended Gaussian image (EGI)** has been proposed [Horn 84]; it is a distribution of surface normals on a Gaussian sphere. All surface normals are moved to the origin and positional information is ignored. A model of an 3D object is then a set of EGIs, each from a different viewing direction on the viewing sphere. The 3D recognition problem is reduced to a set of 2D problems similar to Goad's algorithm (Section 10.3.2). The principal limitation is that only convex objects can be uniquely represented using EGIs. If the connectivity information about the original surface normals is preserved [Liang and Todhunter 90], this convexity can be overcome.

Another similar surface representation is called the **Simplex Angle Image** (SAI) [Higuchi et al. 95]; this stores various surface curvature measures in an generic representation. Starting with a predefined 3D mesh (e.g. an ellipsoid), the mesh is iteratively deformed until it fits the surface. The one-to-one relation between nodes of the initial and deformed meshes is preserved. Curvature features computed from the mesh are stored in the initial mesh and used to represent an object. Connectivity among mesh nodes is stored, and thus the representation copes with non-convex images.

10.4 2D view-based representations of a 3D scene

10.4.1 Viewing space

Most 3D objects or scenes representations discussed hitherto have been **object-centered** – another option is to use **viewer-centered** representations where the set of possible appearances of a 3D object is stored as a collection of 2D images. The trouble is that there is potentially an infinite number of possible viewpoints that induce an infinite number of object appearances. To cope with the huge number of viewpoints and appearances it is necessary to sample a viewpoint space and group together similar neighboring views. The original motivation was the recognition of polyhedral objects, which was later generalized to view-based recognition of curved objects [Ullman and Basri 91]. More recent is the view-based representation of 3D scenes for display from any viewpoint [Beymer and Poggio 96].

Consider two models of the viewing space as a representation of possible views on the object or scene. The general model of the viewing space considers all points in a 3D space in which the 3D object is located at the origin. This viewpoint representation is needed if perspective projection is used. A simplified model is a **viewing sphere** model that is often used in the orthographic projection case. Then the object is enclosed by a unit sphere; a point on the sphere's surface gives a viewing direction. The surface can be densely discretized into view patches.

To simplify working with a viewing sphere it is often approximated by a regular polyhedron, of which the most common choice [Horn 86] is an icosahedron (with 20 equilateral triangular faces. Twenty viewing directions defined by the centers of the triangles are often not enough, in which case the faces are further regularly divided into four triangles in a recursive manner. This yields 80, 320, 1280, ... viewing directions.

10.4.2 Multiview representations and aspect graphs

Other representation methods attempt to combine all the viewpoint specific models into a single data structure. One of them is the **characteristic view technique** [Chakravarty and Freeman 82] in which all possible 2D projections of the convex polyhedral object are grouped into a finite number of topologically equivalent classes. Different views within an equivalence class are related via linear transformations. This general purpose representation specifies the 3D structure of the object. A similar approach is based on **aspect** [Koenderink and van Doorn 79], which is defined as the topological structure of singularities in a single view of an object – aspect has useful invariance properties. Most small changes in vantage point will not affect aspect, and such vantage points (that is, most) are referred to as **stable**. Then the set of stable vantage points yielding a given aspect is contiguous in space, and the

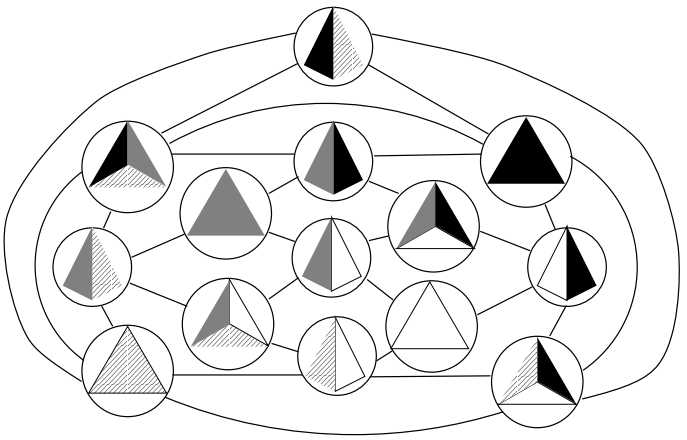


Figure 10.24: *Aspect graph for the tetrahedron.*

space surrounding an object may be partitioned into subspaces from within which the same aspect is generated. An example of the aspect graph for the simplest regular polyhedron – a tetrahedron – is shown in Figure 10.24. Moving from one such subspace into another generates an **event** – a change in aspect; the event is said to connect the two aspects. It is now possible to construct an **aspect graph** with respect to this connectivity relationship in which nodes are aspects and edges are events. This aspect graph is sometimes referred to as the **visual potential** of the object, and paths through it represent orbits of an observer around the object.

10.4.3 Geons as a 2D view-based structural representation

The psychologist Biederman influenced 3D computer vision by the recognition by components theory of human object recognition [Biederman 87]. He advocates structural representations in which shape is composed from a set of primitive parts. Representations used by Biederman are qualitative as opposed to the more commonly used quantitative representations of solids that are specified in terms of numerical parameters. For visual recognition tasks the quantitative description might contain redundant detail while examining some qualitative features of the segmented primitives can ease recognition. Biederman calls 'primal access' the real time entry level shape classification by humans and replication of this ability in machines is the main challenge for geons. However, qualitative representations cannot in general be used to synthesize an image of an object.

Psychophysical experiments provide two main advantages of recognition by components:

- The basic-level representation derived from an image is invariant to scale and translation or orientation in depth.
- The object/scene is composed of parts and this decomposes a complicated task into simpler ones. The information about the scene is given by mutual relations between parts.

Biederman developed a catalogue of three dozen qualitative volumetric primitives called

geons (GEOmetrical iONs). Each member of the catalogue has a unique set of four qualitative distinctions motivated by generalized cylinders:

- Edge; straight or curved.
- Symmetry; rotational, reflective, asymmetric.
- Size variation; constant, expanding, expanding/contracting.
- Spine; straight or curved.

The Cartesian product of values of these distinctions gives the final number of geons. Geons extend the idea of generalized cylinders by adding a qualitative taxonomy; 3D objects would be composed of a number connected geons. Recent views on recognition by component theory can be found in [Dickinson 97].

Biederman proposes an edge-based procedure for segmenting line drawing like images into their geon components – places where geons join are sought using non-accidental alignments and concavities. On the other hand the lack of quantitative information limits the use of geons since distinguishing qualitatively similar objects (for example, parts which differ in scale) becomes difficult. One solution is to add some quantitative information. In general it is possible to recognize an object's type using qualitative information but it would be much more difficult to determine position if no quantitative information were available.

There is still some interest in geons in the vision community. The main drawbacks are:

- It is very difficult to extract a good line drawing from real images. Thus geons have been applied to very constrained domains in which line segmentation is often performed manually, or the objects are unrealistically simple.
- Missing depth information in drawings causes problems in understanding the 3D world.

10.4.4 Visualizing 3D real-world scenes using stored collections of 2D views

Methods which are able to capture a real object and render it from an arbitrary viewpoint usually use a 3D geometric model of the object. The bottleneck of these methods is the 3D reconstruction, which is a non-trivial problem, often failing for objects of more complex shapes. Virtual reality systems can augment real objects to the generated scenes.

Alternatively, the **image-based scene representation paradigm** (or view-based) attempts to display a real 3D scene from any viewing direction without using a 3D model. The scene is represented by a collection of 2D **reference views** instead of a full model. The actual image to be displayed is called a **virtual view** and is created by interpolation from the reference views using correspondences among them. The new bottleneck becomes the correspondence problem, being simpler than 3D reconstruction. The aim of such a procedure is to avoid the difficult problem of consistent 3D model reconstruction and thus more complex objects can be handled. In addition, faster access to a view can be achieved than by rendering the 3D model.

To succeed, the following problems must be solved:

Correspondence problem. How are correspondences between reference views found?

Image interpolation/extrapolation. Given the positions and intensities in reference views, how are they predicted in a new view?

Geometry. How is the visibility of points in the new view determined? It can be shown that without knowledge of geometry it is not possible to render scene correctly.

View selection. How is the optimal set of necessary reference views determined? A solution has been proposed [Werner et al. 96] in which the natural criterion of optimality is the minimal number of views allowing synthesis of an image that looks similar to that which would be seen when looking from the same viewpoint. It is non- trivial to determine a good measure of image similarity that corresponds well to a human understanding of image similarity.

The image based approach copes with complicated free form surfaces – see Figure 10.25. Notice the mistake in the interpolated image in Figure 10.25b – due to a mismatch in correspondences, one of the lines in the top center is doubled. This error is likely to be overlooked, demonstrating how human understanding of an image content is not sensitive to such errors.

(a) (b) (c)

Figure 10.25: *Left and right are two reference views of a linen towel with view directions* 10° *apart. The virtual image in the middle was obtained using interpolation. Courtesy T. Werner. Czech Technical University, Prague.*

It has been thought [Laveau and Faugeras 94, Seitz and Dyer 95, Werner et al. 95, Kumar et al. 95, Irani et al. 95] that displaying a 3D scene from stored 2D images is quite different from rendering a 3D model. The difference seems to follow from Ullman's observation [Ullman and Basri 91] that objects could be recognized just on the basis of linear combination of corresponding points in their orthographic images. This is in contrast to recognition based on verification of a 3D model projected to an image.

Ullman's approach has attracted new attention since it has been shown that for a perspective camera a trilinear function replaces the linear one [Shashua 93]; more recently, it has been shown that for visualization any projective reconstruction of the scene suffices [Laveau and Faugeras 94, Hartley 95], and hence tedious calibration of the camera can be avoided.

It has been shown [Seitz and Dyer 95] that visualizing an object by interpolating close views is a well posed process, and therefore for certain limited tasks a perfect correspondence algorithm is not needed. Elsewhere [Werner et al. 95, Cox et al. 92] it has been shown

that even quite complicated scenes can be shown by interpolating between reference views. However, to make the visual effect realistic, many reference images may be needed [Werner et al. 95]. This is caused by a principal deficiency of image interpolation – its inability to display a general object from an arbitrary viewing angle using the images and the correspondences obtained from a sparse set of views. Surprisingly, no object, not even a convex polyhedron, can be completely viewed by interpolating between a finite number of reference views.

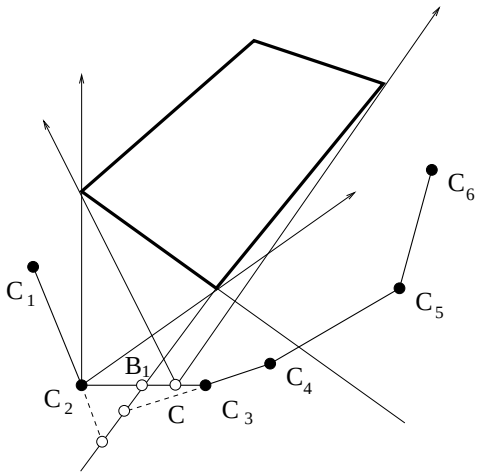


Figure 10.26: *The virtual view C cannot be constructed by interpolation from the views C_2 and C_3, but can be extrapolated from C_3 and C_4.*

Consider the situation in which reference views are located around a simple polyhedron (see Figure 10.26). The images from the virtual camera C lying in the segment B_1C_3 cannot be constructed by interpolating reference views C_2 and C_3 since the camera C_2 does not see both sides of the polyhedron which are seen by the camera C. It will not help to move one of the reference cameras, e.g. C_3, closer to B_1 since then the problem moves to the segment C_3C_4. The only solution would be to increase the density of views near B_1 to infinity: Indeed, the algorithm for finding the best sparse set of reference views [Werner et al. 96] has tended to select many reference views in places where aspect changes – point B_1 is such a place in Figure 10.26.

This argument shows that views can not be constructed by interpolating reference views from different aspects. On the other hand, it is certainly possible to construct view C from C_3 and C_4 by extrapolation of views. The significant difference between interpolation and extrapolation manifests itself on the border between aspects, where the virtual camera has to switch from the views C_1, C_2 to the views C_3, C_4. Unlike in the interpolation case, where switching has been done at the centers of the reference cameras, here it is not clear where to switch between the reference views until the aspect of the object is not known. Finding the changes in aspect is equivalent to finding depth discontinuities; moreover, for reference views in general positions, it is not possible to move smoothly along the object just by linear extrapolation and switching on the borders between aspects as the line C_1C_2 need not intersect the line C_3C_4 on the change of aspect. When crossing the boundary of an aspect, it is also necessary to determine the visibility of the points, because not all of them have to disappear at the same time.

Finding discontinuities and resolving visibility are problems known from 3D surface reconstruction and visualization of 3D models. In order to synthesize the images from a virtual camera moving smoothly around an object, one has to step back from pure image based scene representation and interpolation mechanisms to partial projective reconstructions of the shape and their correct visualization.

A method for constructing virtual views irrespective of the aspect of the reference views has been developed [Werner et al. 97], where a practically technology that permits the building of an image based representation of a 3D scene routinely without human aid is proposed. The novel contribution is:

- The key step towards view extrapolation is to solve the correspondence problem. The proposed solution tracks edge features in a dense sequence of images.
- The visibility of points in virtual views is an important issue. Oriented projective geometry is used to formulate and solve the problem.
- It is shown that it is not necessary to transfer each point from reference views when creating the virtual image. Instead, it is proposed to triangulate the correspondences in reference images, to transfer only their vertices, and to fill triangle interiors.

We demonstrate the displaying of a 3D object of a ceramic doll – see Figure 10.27. The inputs are just two reference images that are captured from two view directions 10^o apart. Figure 10.27e shows a virtual view from the same direction as one of the reference views; it can be seen that the rendering of the triangulated surface gives very similar results. Figures 10.27f, g demonstrate that virtual views can be extrapolated quite far from the two originals. The limit of this range of views is the viewing directions where there is not enough information in the reference images to solve the visibility. This is demonstrated in Figure 10.27h.

The original aim was to visualize a scene using the image-based approach, but it turns out that the whole chain from acquisition to representation to rendering can be made to work. Things start to resemble working with a 3D model; for example, considerations about visibility had to lean on the concept of 3D model, and techniques such as z-buffering, known from 3D model rendering 3D could be used. The boundary separating approaches based on images and 3D models is not sharp, and one could think that this boundary is determined by whether the points in the rendered image are obtained by transfer or reprojection. However, this criterion cannot be taken seriously for there is little difference between transfer and projective reconstruction $\rightarrow$ reprojection. There are three domains which we should distinguish in the spectrum of approaches: (i) view interpolation, (ii) view extrapolation, and (iii) displaying a consistent 3D model.

For further considerations it is necessary to divide the possible viewpoints around the scene (i.e. centers of a camera viewing the scene) into **visibility classes** as follows: A set of points belongs to the same visibility class if and only if the same portion of the scene is visible from any of these points. Note that only convex polyhedra have a finite number of visibility classes.

If two reference cameras are both in the same visibility class and the correspondences satisfy the ordering constraint, view interpolation is a well-posed problem [Seitz and Dyer 95] and solving visibility can be neglected. When the reference views are not in the same visibility class, the interpolation yields a good approximation of the correct view if the reference views

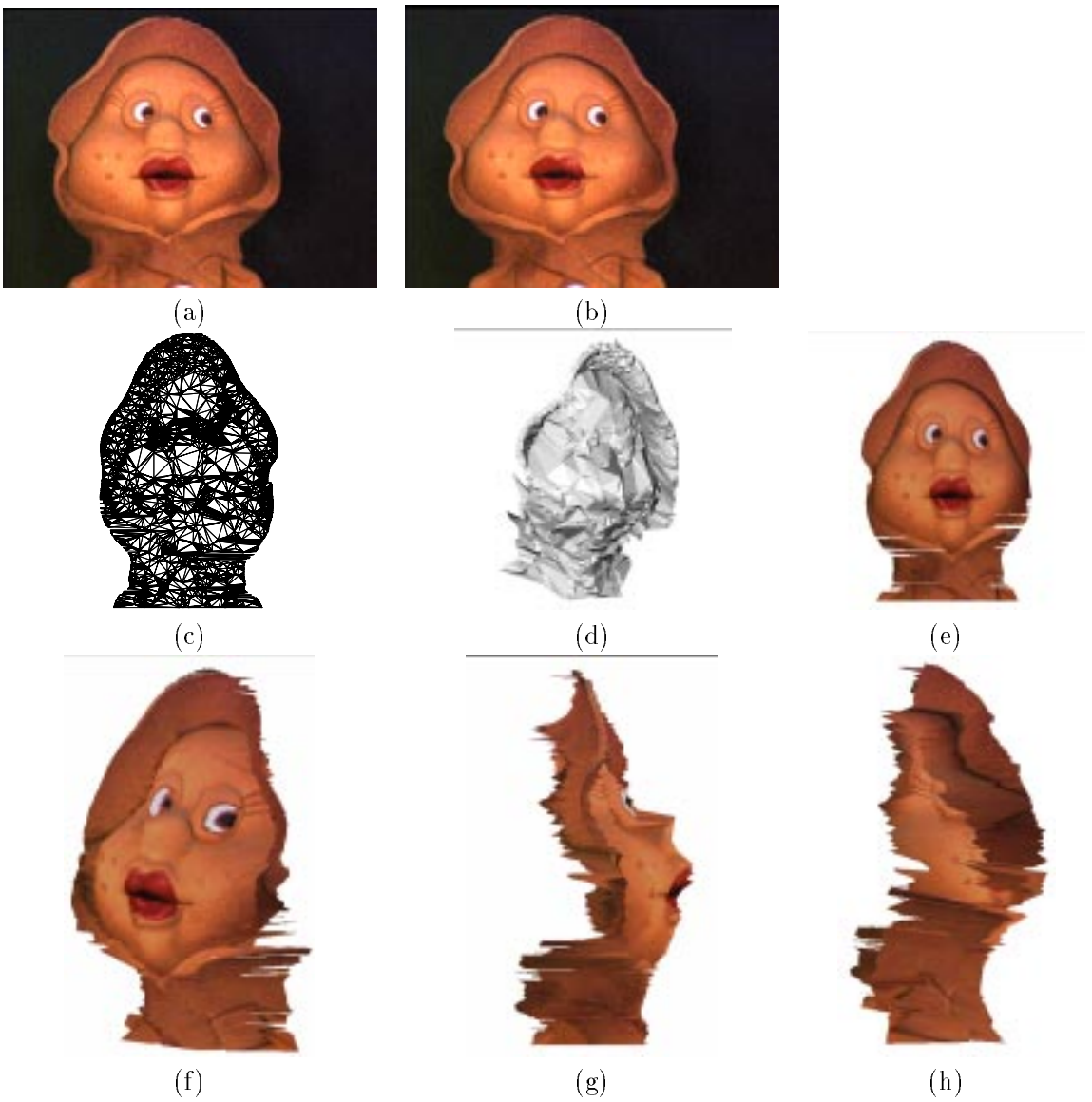


Figure 10.27: *The results for the second object: (a) and (b) the reference views; (c) the triangulation of the first reference view; (d) the projective model; (e) the virtual view from the same viewpoint as (a); (f) and (g) other virtual views; (h) shows the incorrectly solved visibility in (g) – the extrapolation extended beyond the reasonable limit (there was an attempt to look at something which was not visible in the reference images). Courtesy T. Werner. Czech Technical University, Prague.*

are close to each other. Moreover, for close reference views, linear interpolation [Werner et al. 95] can replace the general transfer based on the trilinear relationship.

It can be shown that no scene can be displayed correctly from an **arbitrary** viewpoint by switching among view interpolation algorithms used for different sets of reference views. However, if both reference views and the virtual cameras are all in the same visibility class, view extrapolation remains also well-posed. The extrapolation becomes in this case as simple as the interpolation because no visibility has to be solved. Convex polyhedra can be displayed correctly from any viewpoint by switching among these algorithms used for the different sets of reference views.

If the virtual camera is in a different visibility class than the reference ones, this simple algorithm cannot be used and visibility in the virtual image plane must be solved explicitly. Since determining whether the virtual camera is in the same class as the reference ones is very difficult it is in practice always necessary to solve visibility. Actually, this is exactly the situation considered here which is called view extrapolation.

Some views of complex objects cannot be obtained from the reference cameras placed in the same visibility class, because they have in general infinitely many classes. In such a case, only part of the virtual view can be obtained correctly from any set of reference views. The whole virtual view needs to be constructed by **fusing** parts obtained from different sets of reference views. Surprisingly, it seems that there is no qualitative difference between fusing parts of images to get a physically valid virtual view and the reconstruction of a consistent projective model.

10.5 Summary

- This material is overview in nature. We have given a taxonomy of various 3D vision tasks, an overview of current approaches, formulated tasks, shown some applications, and hinted at recent research directions.
- **Shape from X**
  - Shape may be extracted from motion, optical flow, texture, focus/defocus, vergence and contour.
  - Each of these techniques may used to derive a 2.5D sketch for Marr's vision theory; they are also of practical use on their own.
- **Full 3D objects**
  - Line labeling is an outmoded but accessible technique for reconstructing object with planar faces.
  - Transitions to 3D objects need a co-ordinate system that is object centered.
  - 3D objects may be measured mechanically or by computed tomography.
  - Volumetric modeling strategies include constructive solid geometry, superquadrics and generalized cylinders.
  - Surface modeling strategies include boundary representations, triangulated surfaces and quadric patches.

- **3D model-based vision**
  - To create a full 3D model from a set of range images the measured surfaces must first be registered – rotations and translations should be found that match one surface to another.
  - Model-based vision uses a priori knowledge about an object to ease its recognition.
  - Goad's algorithm is an illustration that searches for polyhedra in a single intensity image.
  - Techniques exist to locate curved objects from range images.
- **2D view-based representations of a 3D scene**
  - 2D view-based representations of 3D scenes may be achieved with multiview representations or geons.
  - It is possible to select a few stored reference images, and render any view from them.
  - Interpolation of views is not enough and view extrapolation is needed. This requires knowledge of geometry and the view-based approach does not differ significantly from 3D geometry reconstruction.

10.6 Exercises

Short-answer questions

1. For each of the following *shape from X* techniques, briefly describe the principle underlying it, explain how they can be of practical use, and give some examples.
   - motion
   - texture
   - focus
   - vergence
   - contour
2. Give a counter example in which apparent motion measured by an optical flow field does not correspond to actual motion of the 3D points. (Hint: consider an opaque sphere; (1) the sphere does not move and illumination changes, (2) the sphere rotates and the illumination is constant.)
3. Fashion designers cheat the human ability to derive shape from texture. How do they do it?
4. Give an example of a rim which is not a planar curve.
5. Are there line drawings that do not correspond to physically plausible polyhedral objects? If yes, show an example.
6. What was the contribution of image interpretation via line labeling to computer vision? Where else can it be used and why? (Hint: model-based 3D object recognition)
7. Draw a picture that functionally decomposes the human body into several generalized cylinders from the point of view of movement.
8. Describe how to create a full 3D model of an object from range images taken from different views.
9. Explain the principle of 2D view-based representation of a 3D scene.

Problems

1. Conduct a laboratory experiment that establishes depth from focus. Take a lens with small depth of focus – if you have an ordinary lens at hand open the aperture as much as possible. Mount the camera with the lens on a stand that can adjust the distance between the camera and the object – a photo magnifier stand can be used for this purpose).

   Select a window in the image that corresponds to a point in the scene from which the distance is measured. Devise an algorithm that assists in achieving the sharpest image. (Hint: the sharpest image has the most high frequencies; is there some method to find such an image that is simpler than the Fourier transform?)

   Finally, the distance can be read from the mechanical scale on the stand.

2. Write an essay about shape. Most computer vision techniques measure co-ordinates of distinct points on a 3D object surface, e.g. those where correspondences were found; the result is a 3D point cloud. The human notion of shape is not of a point cloud. Elaborate on this problem, e.g. in the case of human face where 3D data were measured using stereopsis. How would you move from a point cloud to a surface? Is the surface the shape we seek? (Hint: Shape is treated from the geometer's point of view in [Koenderink 90]).

10.7 References

[Aloimonos 93] Y Aloimonos, editor. *Active Perception*. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1993.

[Aloimonos and Swain 85] J Aloimonos and M J Swain. Shape from texture. In *Proceedings of the Ninth Joint Conference on Artificial Intelligence IJCAI*, volume 2, pages 926–931, 1985.

[Arman and Aggarwal 93] F Arman and J K Aggarwal. Model-based object recognition in dense-range images — A review. *ACM Computing Surveys*, 25(1):5–43, 1993.

[Bajcsy and Lieberman 76] R Bajcsy and L Lieberman. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5:52–67, 1976.

[Barr 81] A H Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, January 1981.

[Besl and Jain 85] P J Besl and R C Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, March 1985.

[Besl and McKay 92] P J Besl and N D McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[Beymer and Poggio 96] D Beymer and T Poggio. Image representations for visual learning. *Science*, 272:1905–1909, 1996.

[Biederman 87] I Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[Binford 71] T O Binford. Visual perception by computer. In *Proceedings of the IEEE Conference on Systems, Science and Cybernetics*, IEEE, Miami, Fl, December 1971.

[Blake and Yuille 92] A Blake and A Yuille. Active vision. In *MIT Press*, 1992.

[Blostein and Ahuja 89] D Blostein and N Ahuja. Shape from texture: Integrating texture-element extraction and surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1233–1251), 1989.

[Bowyer 92] K W Bowyer, editor. *Special issue on directions in CAD-based vision*, volume 55, 1992.

[Brady 84] M Brady. Representing shape. In M Brady, L A Gerhardt, and H F Davidson, editors, *Robotics and Artificial Intelligence*, pages 279–300. Springer + NATO, Berlin, 1984.

[Bray and Hlavac 91] A J Bray and V Hlavac. Properties of local geometric constraints. In *Proceedings of the British Machine Vision Conference*, pages 95–103, Glasgow, U.K., September 1991.

[Brooks et al. 79] R A Brooks, R Greiner, and T O Binford. The ACRONYM model-based vision system. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-6, Tokyo*, pages 105–113, 1979.

[Chakravarty and Freeman 82] I Chakravarty and H Freeman. Characteristic views as a basis for three-dimensional object recognition. *Proceedings of The Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, 336:37–45, 1982.

[Chen and Kak 89] C H Chen and A C Kak. A robot vision system for recognizing 3D objects in low order polynomial time. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1535–1563, 1989.

[Clocksin 80] W F Clocksin. Perception of surface slant and edge labels from optical flow – a computational approach. *Perception*, 9:253–269, 1980.

[Clowes 71] M B Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[Cox et al. 92] I J Cox, S Hingorani, B M Maggs, and S B Rao. Stereo without disparity gradient smoothing: A Bayesian sensor fusion solution. In *British Machine Vision Conference*, pages 337–346, Springer-Verlag, Berlin, 1992.

[Dickinson 97] S J et al. Dickinson. Panel report: the potential of geons for generic 3-D object recognition. *Image and Vision Computing*, 15:277–292, 1997.

[Faugeras 93] O Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, 1993.

[Faugeras and Hebert 86] O D Faugeras and M Hebert. The representation, recognition, locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[Flynn and Jain 92] P J Flynn and A K Jain. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP – Image Understanding*, 55(2):119–129, 1992.

[Goad 86] C Goad. Fast 3D model-based vision. In A P Pentland, editor, *From Pixels to Predicates*, pages 371–374. Ablex Publishing Corporation, Norwood, NJ, 1986.

[Grimson 89] W E L Grimson. On the recognition of curved objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6), 1989.

[Gubais et al. 92] L J Gubais, D E Knuth, and Sharir M. Randomized incremental construction of Delunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[Hartley 95] R I Hartley. A linear method for reconstruction from lines and points. In *Proceedings of the 5th International Conference on Computer Vision*, pages 882–887, IEEE Computer Society Press, Boston USA, June 1995.

[Hayes-Roth et al. 83] F Hayes-Roth, D A Waterman, and D B Lenat. *Building Expert Systems*. Addison-Wesley, Reading, Ma, 1983.

[Higuchi et al. 95] K Higuchi, M Hebert, and K Ikeuchi. Building 3-D models from unregistered range images. *Graphics Models and Image Processing*, 57(4):313–333, July 1995.

[Hlaváč et al. 94] V Hlaváč, T Pajdla, and M Sommer. Improvement of the curvature computation. In *Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem, Israel*, volume 1, Computer Vision and Image Processing, pages 536–538. IEEE, October 1994.

[Hoppe et al. 92] H Hoppe, T DeRose, T Duchamp, J McDonald, and W Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[Horn 84] B K P Horn. Extended Gaussian image. *Proceedings of the IEEE*, 72(12):1671–1686, December 1984.

[Horn 86] B K P Horn. *Robot Vision*. MIT Press, Cambridge, Ma, 1986.

[Huffman 71] D A Huffman. Impossible objects as nonsense sentences. In B Metzler and D M Michie, editors, *Machine Intelligence*, volume 6, pages 295–323. Edinburgh University Press, 1971.

[Irani et al. 95] M Irani, P Anandan, and S Hsu. Mosaic-based representations of video sequences and their applications. In *Proceedings of the 5th International Conference on Computer Vision*, pages 605–611. IEEE Computer Society Press, May 1995.

[Jaklič 97] A Jaklič. *Construction of CAD models from Range Images*. PhD thesis, University of Ljubljana, Department of Computer and Information Science, Ljubljana, Slovenia, March 1997.

[Kanatani and Chou 89] K Kanatani and T C Chou. Shape from texture: General principle. *Artificial Intelligence*, 38(1):1–48, 1989.

[Klir 91] G J Klir. *Facets of System Science*. Plenum Press, New York, USA, 1991.

[Koenderink 90] J J Koenderink. *Solid Shape*. The MIT Press, 1990.

[Koenderink and van Doorn 79] J J Koenderink and A J van Doorn. Internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32(4):211–216, 1979.

[Kriegman 92] D J Kriegman. Computing stable poses of piecewise smooth objects. *CVGIP – Image Understanding*, 55(2):109–118, 1992.

[Krotkov 87] E Krotkov. Focusing. *International Journal of Computer Vision*, 1:223–237, 1987.

[Krotkov and Bajcsy 93] E P Krotkov and R Bajcsy. Active vision for reliable ranging: Cooperating focus, stereo, and vergence. *International Journal of Computer Vision*, 11(2):187–203, October 1993.

[Kumar et al. 95] R Kumar, P Anadan, M Irani, J Bergen, and K Hanna. Representation of scenes from collection of images. In *Proceedings of the Visual Scene Representation Workshop, Boston, MA., USA, June 24*, pages 10–17. IEEE Computer Society Press, 1995.

[Laveau and Faugeras 94] S Laveau and O Faugeras. 3-D scene representation as a collection of images. In *Proc. of 12th International Conf. on Pattern Recognition, Jerusalem, Israel*, pages 689–691, October 9–13 1994.

[Leonardis et al. 97] A Leonardis, A Jaklič, and F Solina. Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1289–1295, November 1997.

[Liang and Todhunter 90] P Liang and J S Todhunter. Representation and recogntion of surface shapes in range images: A differential geometry approach. *Computer Vision, Graphics, and Image Processing*, 52:78–109, 1990.

[Lowe 85] D G Lowe. *Perceptual Organisation and Visual Recognition*. Kluwer Nijhoff, Norwell Ma, 1985.

[Lowe 89] D G Lowe. Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, 1:119–130, 1989.

[Malik and Maydan 89] J Malik and D Maydan. Recovering three-dimensional shape from a single image of curved object. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 11(6):555–566, June 1989.

[Marr 82] D Marr. *Vision – A Computational Investigation into the Human Representation and Processing of Visual Information.* W H Freeman and Co., San Francisco, 1982.

[Mokhtarian 95] F Mokhtarian. Silhouette-based object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:539–544, 1995.

[Nayar and Nakagawa 94] S K Nayar and Y Nakagawa. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994.

[Nayar et al. 96] S K Nayar, M Watanabe, and M Hoguchi. Real-time focus range sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1197, December 1996.

[Negahdaripour and Horn 85] S Negahdaripour and B K P Horn. Determining 3D motion of planar objects from image brightness measurements. In *Proceedings of the Ninth Joint Conference on Artificial Intelligence IJCAI*, volume 2, pages 898–901, 1985.

[Nevatia et al. 94] R Nevatia, M Zerroug, and F Ulupinar. Recovery of three-dimensional shape of curved objects from a single image. volume 2, pages 101–129, Academic Press, San Diego, USA, 1994.

[Newman et al. 93] T S Newman, P J Flynn, and A K Jain. Model-based classification of quadric surfaces. *CVGIP: Image Understanding*, 58(2):235–249, September 1993.

[Pajdla and Van Gool 95] T Pajdla and L Van Gool. Matching of 3-D curves using semi-differential invariants. In *Proceedings of the 5th International Conference on Computer Vision*, pages 390–395. IEEE Computer Society Press, May 1995.

[Pentland 87] A Pentland. A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):523–531, 1987.

[Ponce et al. 92] J Ponce, A Hoogs, and D J Kriegman. On using CAD models to compute the pose of curved 3d objects. *CVGIP – Image Understanding*, 55(2):184–197, 1992.

[Preparata and Shamos 85] F P Preparata and M I Shamos. *Computational Geometry - An Introduction.* Springer Verlag, Berlin, 1985.

[Quarendon 84] P Quarendon. *WINSOM User's Guide.* IBM, IBM UK Scientific Centre, August 1984.

[Rissanen 89] J Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Co., Series in Computer Science, IBM Almaden Research Center, San Jose, California U S A, 1989.

[Rivlin and Weiss 95] E Rivlin and I Weiss. Local invariants for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):226–238, March 1995.

[Roberts 65] L G Roberts. Machine perception of three-dimensional solids. In J T Tippett, editor, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, Ma, 1965.

[Sato et al. 92] K Sato, K Ikeuchi, and T Kanade. Model based recognition of specular objects using sensor models. *CVGIP – Image Understanding*, 55(2):155–169, 1992.

[Seales and Dyer 92] W B Seales and C R Dyer. Viewpoints from occluding contour. *CVGIP – Image Understanding*, 55(2):198–211, 1992.

[Seitz and Dyer 95] S M Seitz and C R Dyer. Physically-valid view synthesis by image interpolation. In *Proceedings of the Visual Scene Representation Workshop, Boston, MA., USA, June 24*, pages 18–27. IEEE Computer Society Press, 1995.

[Shashua 93] A Shashua. On geometric and algebraic aspects of 3D affine and projective structures from perspective 2D views. Technical Report "AI Memo No. 1405, Massachusetts Institute of Technology, Artifical Intelligence Laboratory, Massachusetts Institute of Technology, Artifical Intelligence Laboratory, Cambridge, MA, USA, July 1993.

[Shomar and Young 94] W J Shomar and T Y Young. Three-dimensional shape recovery from line drawings. In *Handbook of Pattern Recognition and Image Processing: Computer Vision*, volume 2, pages 53–100, Academic Press, San Diego, USA, 1994.

[Solina and Bajcsy 90] F Solina and R Bajcsy. Recovery of parametric models from range images: the case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, December 1990.

[Soroka and Bajcsy 78] B I Soroka and R K Bajcsy. A program for describing complex three dimensional objects using generalised cylinders. In *Proceedings of the Pattern Recognition and Image Processing Conference (Chicago)*, pages 331–339, IEEE, New York, 1978.

[Soucy and Laurendeau 96] M Soucy and D Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, 63(1):1–14, January 1996.

[Stevens 79] K A Stevens. Representing and analyzing surface orientation. In P A Winston and R H Brown, editors, *Artificial Intelligence: An MIT Persepctive*, volume 2. MIT Press, Cambridge, Ma, 1979.

[Sugihara 86] K Sugihara. *Machine interpretation of line drawings.* MIT Press, Cambridge, MA, USA, 1986.

[Terzopoulos and Fleischer 88] D Terzopoulos and K Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[Terzopoulos and Metaxas 91] D Terzopoulos and D Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[Terzopoulos et al. 88] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3-D shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[Ullman 79] S Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, Ma, 1979.

[Ullman 96] S Ullman. *High-level Vision: Object Recognition and Visual Cognition.* A Bradford Book. MIT Press, Cambridge, Ma., USA, 1996.

[Ullman and Basri 91] S Ullman and R Basri. Recognition by linear combination of models. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 13(10):992–1005, October 1991.

[Uray 97] P Uray. *From 3D point clouds to surface and volumes.* PhD thesis, Technische Universitaet Graz, Austria, October 1997.

[Šára and Bajcsy 98] R Šára and R Bajcsy. Fish-scales: Representing fuzzy manifolds. Accepted to International Conference on Computer Vision to be held in New Delhi, India., January 1998.

[Waltz 75] D L Waltz. Understanding line drawings of scenes with shadows. In P H Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw Hill, New York, 1975.

[Weiss 88] I Weiss. Projective invariants of shapes. In *Proceedings of Image Understanding Workshop*, volume 2, pages 1125–1134, Cambridge, Ma, 1988.

[Weiss and Rosenfeld 96] I Weiss and A Rosenfeld. A simple method for ellipse detection. Technical Report CS-TR-3717, University of Maryland, Center for Automation Research, College Park, MD, USA, 1996.

[Werner et al. 95] T Werner, R D Hersch, and V Hlaváč. Rendering real-world objects using view interpolation. In *Proceedings of the 5th International Conference on Computer Vision*, pages 957–962, IEEE Press, Boston, USA, June 1995.

[Werner et al. 96] T Werner, V Hlaváč, A Leonardis, and T Pajdla. Selection of reference views for image-based representation. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume I – Track A: Computer Vision, pages 73–77, IEEE Computer Society Press, Los Alamitos, CA., USA, Vienna, Austria, August 1996.

[Werner et al. 97] T Werner, T Pajdla, and V Hlaváč. Visualizing 3-D real-world scenes using view extrapolation. Technical Report K335-CMP-1997-137, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 12135 Praha, Czech Republic, June 1997.

[Witkin 81] A P Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.

[Worring and Smeulders 93] M Worring and A W M Smeulders. Digital curvature estimation. *CVGIP: Image Understanding*, 58(3):366–382, November 1993.