10. SURFACE RECOVERY THROUGH DEPTH COMPUTATION

In computer vision (as in human vision) the input images which convey 3-dimensional information are themselves 2-dimensional. Examples of 3D information that we may wish to extract from images include: 3D shape of a surface, distance of the a surface from an observer, depth of points in a scene.

There is a convention in computer vision that "3D" shape refers to *entire* object, whereas "2.5D" shape refers to a part of the object which can be seen from a given viewpoint.

Computer vision techniques for recovering of 3D (or, strictly speaking, 2.5D) information from images are usually inspired by human abilities and are using the cues thought to contribute to the human depth perception. The main schemes are:
- depth from stereo images
- surface (and depth) from shading
- depth from texture
- depth from optical flow

All the methods exploit the idea that if the depth for each point on a surface is known, the entire (2.5D) surface can be reconstructed. The problem is, how to measure depth from a grey scale image or images? An alternative is to use the so called *range images* in which pixel intensity is a measure of *distance* between a sensor and a point in the image. Range images can be obtained directly, by using range detectors (e.g. sonar, laser) instead of optical cameras, and are very helpful for 2D and 3D analysis if combined with intensity images.

Stereo images

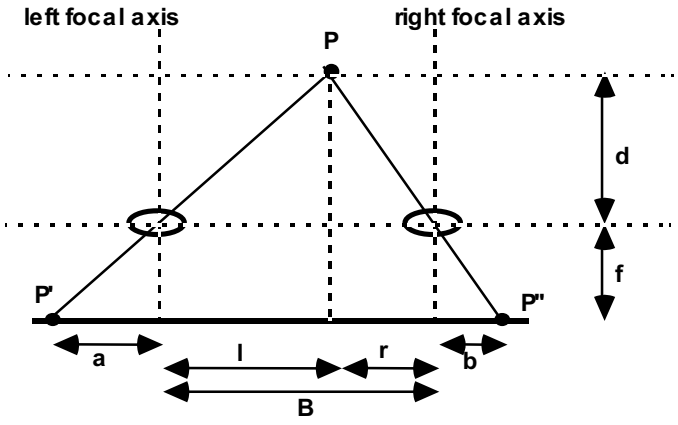
The principle of stereo vision (binocular imaging) is based on geometrical fact that two images taken from two positions can give clues for relative depth measurements if:
- baseline and focal length known
- disparity can be found from the image

The outline of the technique
- take two images separated by a baseline
- match points between two images
- derive two lines (linking each of the "eyes" with a point in space)
- find the lines intersection
- calculate depth

The most difficult part is *matching* points between images.

The scheme illustrated below, mathematically the simplest, assumes that the lines of sight for the two eyes are parallel. More commonly the two eyes converge, and the equation more complicated. There exist many variants of this method, differing in mathematical approach as well as in assumptions they make. For example some recent methods do not need calibration (e.g. Zhang et al). One interesting method proposed by Marr & Poggio (1976) uses connectionist implementation.

© E Claridge, School of Computer Science, The University of Birmingham.



P point on the object
P' left image of the point P
P'' right image of the point P
a left displacementt
b right displacement
B total displacement (B=l+r)
f focal lenght
d distance to be found

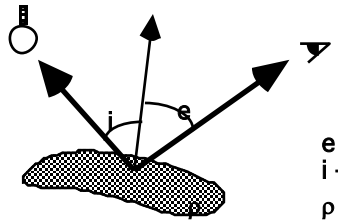
$$\frac{d}{l} = \frac{d+f}{l+a} \qquad \frac{d}{r} = \frac{d+f}{r+b}$$

$$d = fB/(a+b)$$

Shading

Under uniform illumination it is possible to calculate surface orientation for each point (and hence the surface shape) from shading. The method draws on the following facts.

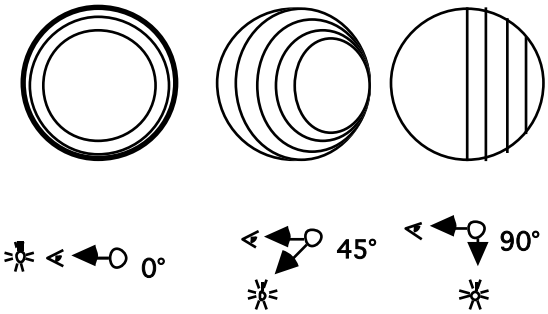
- Light is reflected from a surface in a direction of the viewer.
- Amount of light reflected depends on various angles and the surface properties.
- Brightness of a *Lambertian surface* (ideally mat) depends only on the direction of a light source (the angle it makes with a surface normal) and the surface property.
- Isobrightness lines are the lines along which brightness is the same.
- Isobrightness lines projected onto a flat surface constitute *reflectance maps* (like maps in cartography).
- Surface direction is recovered from shading



e - emergent angle
i - incidence angle
ρ - surface albedo

For Lambertian surface $E = \rho \cos i$

Isobrightness lines on Lambertian spheres



Notes:

© E Claridge, School of Computer Science, The University of Birmingham.

Computational theory
- Determine direction of the light source
- For every surface (point) measure amount of light reflected from the surface in direction of the viewer
- Recover orientation of the surface (point) w.r.t. direction of light source.

Constraints from environment
- For matt surfaces the amount of light reflected depends on the incident angle (cosine law)
- Objects are cohesive and surface gradient changes smoothly (usually)
- For smooth surfaces, the surface normal at the occluding boundary lies in the lane perpendicular to the direction of view.

Algorithm
- Identify boundary
- For each boundary point
  - determine the gradient
  - assign gradient space coordinates
- For each interior point
- get point brightness
- determine iso-brightness line in gradient space
- get gradient space coordinates of the nearest neighbours
- assign gradient space coordinates consistent with the point's brightness and gradient of its nearest neighbours
- Repeat until no changes occur

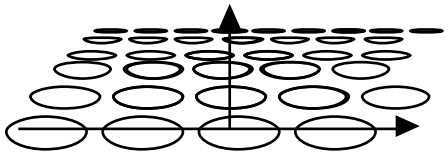
Notes:

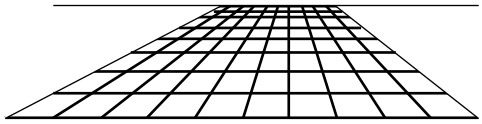
© E Claridge, School of Computer Science, The University of Birmingham.

Texture gradient

It is possible to calculate surface orientation from images of textured surfaces:

- Texture is first segmented into primitives
- The maximum rate of change of projected primitive sizes is the direction of the texture gradient
- The orientation of the direction determines a rotation of the plane about the camera line of sight
- The magnitude of gradient determines how much the plane is tilted with respect to the camera
- Knowing a real shape of a texture primitive, orientation and tilt can be evaluated
- For textures in the form of regular grids a vanishing point can be calculated



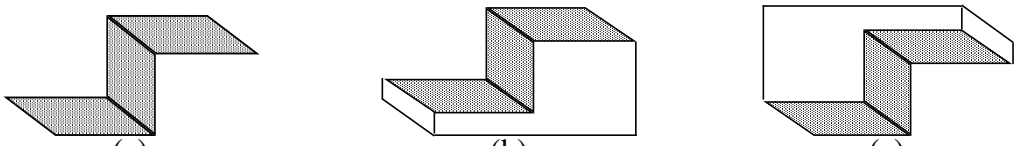


As well as exploiting natural textures, smooth surface can be made to appear textured by projecting onto them a regular pattern. Surface shape can be accurately evaluated from this texture. If shape of a texture primitive is not known or its distribution is irregular, a stereo-pair of textured images can be used for depth reconstruction.

© E Claridge, School of Computer Science, The University of Birmingham.

SURFACE RECOVERY THROUGH EXPLOITING NATURAL CONSTRAINTS

The surface in figure (a) can have two different 3D interpretations depending on the context (shown in figures (b) and (c)). Additional surfaces in (b) and (c) *constrain* interpretation of the surface.



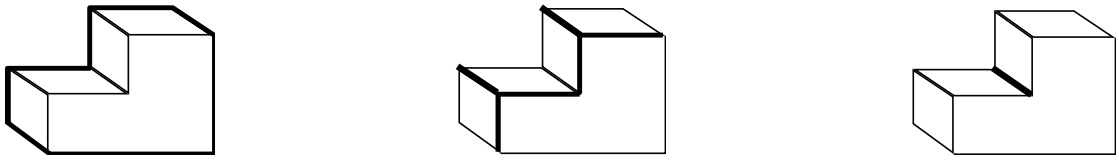
(a) (b) (c)

Polyhedral scene labelling

Vertices in three-dimensional scenes depicting polyhedra (3D objects made of surfaces which are polygons) generate *junctions* in line drawings of the scenes. If there are no shadows or cracks and all vertices are three-faced then only four types of junctions can occur in the drawing:

Ell Arrow Fork Tee

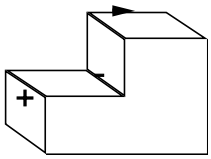
The lines in this type of drawings divide into the following types:

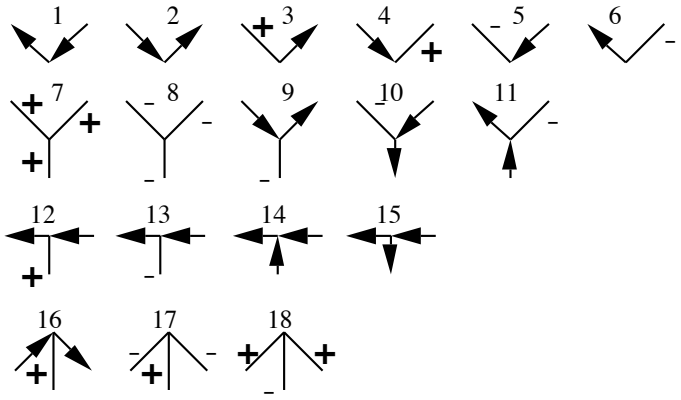


Boundary Interior convex Interior concave

Line types are identified on drawings by line labels as follows:

+ convex lines
- concave lines
> boundary lines





Combinations of line labels surrounding junctions are called *junction labels*. Out of 208 all possible junction labels only 18 occur in reality.

By correctly labelling junctions in line drawings it is possible to derive three-dimensional interpretation of the polyhedral scene and detect 'impossible' polyhedra.

Overall scheme

Create the catalogue of junctions

We look at every possible three-faced vertex form every possible direction. This is done by considering all ways of filling up eight octants of the space and viewing each of the

© E Claridge, School of Computer Science, The University of Birmingham.

resulting vertices from the unfilled octants. The junction catalogue for the restricted class of polyhedra we are disussing is shown above.

Find correct labels

The three-dimensional objects are projected onto two-dimensional surface. We can recover three-dimensional information from this two-dimensional image by imposing geometrical constraints (i.e. law of 3D world and law of projection).We start with external boundary lines. It is relatively easy to find them - they are the ones we 'hit' when traversing an image starting from the image edge. Then we use our knowledge of the possible junctions (as in the catalogue) and the knowledge that line labels must be consistent between the junctions. It could be a tedious process, especially for complex drawings. The constraint propagation is a possible technique.

Labelling through constraint propagation

Constraint propagation techniques use constraints inherent in the domain to which they are applied in order to achieve *global* consistency via *local* computation.

A basic labelling problem can be described as follows:

- Given a finite input (a relational structure of objects), a set of labels, and a set of constraints,
- find a consistent labelling, i.e. assign labels to objects without violating the constraints. Initially each object is given all the legal labels according to unary constraints.
- Constraints involving more and more objects are incorporated successively as each object is checked to see if it is consistent with the new constraint.
- At each step the new object's constraint is propagated, i.e. the structure is checked if it is consistent with the new constraint.

Labelling applied to polyhedral junctions:

- To solve the problem of consistent junction labelling, each junction is initially assigned all the labels it can legally have.
- All the junctions are examined it tern and consistency of labelling between the current junction and all its neighbours is examined.
- If any of the labels is illegal in this context, it is removed and all the neighbouring junctions are checked themselves for consistency.
- This continues until no more changes are occurring.

© E Claridge, School of Computer Science, The University of Birmingham.

Waltz algorithm for propagating symbolic constraints

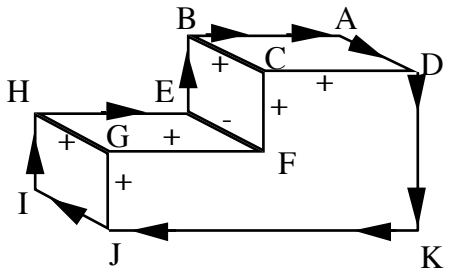
1 Form a queue consisting of all junctions
2 Until queue is empty
  2.1 Remove the first element from the queue; call it a current junction.
    2.1.1 If the current junction has never been visited, create a stack of junction labels for it consisting of all possible junction labels for the junction type involved. Note that the stack change has occurred.
    2.1.2 If any junction label from the current junction's stack is incompatible with all the junction labels in any neighbouring junction's stack, eliminate that incompatible label from the current junction's stack. Note that the stack change has occurred.

  2.2 If a stack change has occurred, for each neighbouring junction with a stack which is not on the queue, add that junction to the front of the queue.

The Waltz algorithm to label the junctions in the line drawing

In this limited world start with the assumption that it is possible to label all the edges lying on the external contour; the junctions associated with them are therefore of type 1.

Step	Junction	Possible labels	Retained labels	Constraint used
1	A	1,2,3,4,5,6	1	external contour junction
2	B	16,17,18	16	'A' junction
etc.				



The labelled polyhedron

Notes:

© E Claridge, School of Computer Science, The University of Birmingham.

Further reading and exploration

Surface recovery from depth computation

Bruce, V. et al, Chapter 7.

Sonka, M. et al, parts of Chapter 9.

Marr, D. & Poggio, T. (1976) Cooperative computation of stereo disparity. Science 194, 283-287.

Horn, B.K.P. (1977) Understanding image intensities. Artificial Intelligence 8, 201-231.

Horn, B.K.P. & Brooks, M.J. (eds) (1989) Shape from Shading: MIT Press.

Ikeuchi, K. & Horn, B.K.P.(1989) Numerical shape from shading and occluding boundaries. In: Shape from Shading (Horn et al eds.), 245-299.

Zhang, Z., Luong, Q.T., Faugeras, O. (1994) Self-calibration of an uncalibrated stereo rig from one unknown motion. Proceedings of the British Machine Vision Association, 1994, 499-508.

Surface recovery through exploiting natural constraints

Bruce, V. et al, Chapter 6, 120-125.

Roberts (1965) Machine perception of 3D solids. Optical and Electro-Optical information processing (Tippet JT Ed): MIT Press, 159-197.

Guzman (1968) De-composition of a visual scene into 3D bodies. AFIPS Conference Proc. 33, 291-304.

Waltz (1975) Understanding line drawings of scenes with shadows. The Psychology of Computer Vision (Winston PH Ed): McGraw Hill, 19-91.

Mackworth (1973) Interpreting pictures of polyhedral scenes. Artificial Intelligence 4, 121-137.

Winston PH (1992) Artificial Intelligence, Chapter 12 “Symbolic constraints and propagation”: Addison-Wesley, Third edition.

© E Claridge, School of Computer Science, The University of Birmingham.