



This is to certify that the

dissertation entitled

Three-Dimensional Scene Representation
using Structured Light

presented by

Gongzhu Hu

has been accepted towards fulfillment
of the requirements for

the PhD degree in Computer Science

George C Stockman
Major professor

Date 20 May 1988



RETURNING MATERIALS:

Place in book drop to
remove this checkout from
your record. FINES will
be charged if book is
returned after the date
stamped below.

300 D 2264

336 1000

--	--	--

**THREE-DIMENSIONAL SCENE REPRESENTATION
USING STRUCTURED LIGHT**

By

Gongzhu Hu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1988

ABSTRACT

3-D SCENE REPRESENTATION USING STRUCTURED LIGHT

By

Gongzhu Hu

This thesis explores general use of structured lighting for surface and object representation. Using structured light, image features are better defined and easier to detect because surface geometry is made explicit through the light stripe networks. Because of the effectiveness in shape inference and its speed and cost potential, the structured lighting approach provides a promising and economical alternative for 3-D machine vision, especially for industrial applications where controlled environment is assumed. Previous work using structured light has been focused on 3-D sensing, but few projects have gone further to include structured light in shape inference, surface representation processes, and scene analysis.

This thesis investigates problems in using structured light from a general point of view. Not only is structured light used in 3-D sensing, but also surface shapes are inferred via both 3-D data fitting and 2-D textures created from the structured lighting. In addition, surface representation is constructed based on the light-striped surface patches and their associated boundary lines. This type of $2\frac{1}{2}$ -D "wing" representation is suitable for object recognition through model matching.

3-D sensing is accomplished by computing 3-D surface solutions at the sparse stripe grid points via triangulation. The stripe identification problem is partially solved using a set of geometric and topological constraints. Surface shapes are inferred by two

independent methods: a) surface fitting of the computed 3-D data followed by a curvature computation and classification, and b) classification via features of 2-D stripe textures. Experiments show that both methods are feasible. A fusion scheme that combines features obtained from a striped image and features extracted from an intensity image of the same scene is employed to construct a $2\frac{1}{2}$ -D representation. The features used in the representation include surface patches (from striped image), edges (primarily from intensity image), and their spatial relationships. A set of inference rules are developed that label these representation primitives. The $2\frac{1}{2}$ -D representation will be delivered to the matching procedure for recognition, which is not studied in this thesis.

Experiments with real images of multiple object scenes provide encouraging results. More robust feature extraction, segmentation of general objects, and modeling and matching approaches are currently under investigation.

ACKNOWLEDGEMENTS

For the completion of this thesis, I would like to thank Professor George Stockman, my thesis advisor, for his valuable guidance and great patience in allowing me to pursue this research. Without his invaluable suggestions, great inspiration and help, this work would have been impossible. Members of my thesis committee, Professor Anil Jain, Professor Carl Page, Professor Eric Goodman, and Professor Richard Hall, provided me with continual encouragement and many excellent comments.

My thanks also go to Professor Richard Dubes, Professor M. Tuceryan and all the members in the Pattern Recognition and Image Processing (PRIP) lab at Michigan State University for their help during my stay in the PRIP lab. The Department of Computer Science, and many friends showed great support and concern.

I also wish to acknowledge the financial support I received during my doctoral studies from NSF Grants MCS-8304011, DCR-8600371 and IRI-8743697, and the Department of Computer science at Michigan State University.

My greatest gratitude is for my parents, Huoxien Hu and Renzhang Yang, for providing me with the best fundamental education, for my wife Shu for her patience and encouragement, and for my daughter Po for being wonderful.

Table of Contents

Table of Figures	viii
Table of Tables	x
Chapter 1 Introduction	1
1.1. 3-D Scene Analysis.....	2
1.2. Problems Addressed in the Dissertation	5
Chapter 2 3-D Interpretation Using Light Striping	8
2.1. Three Components of Visual Processing.....	9
2.1.1. Light Source.....	9
2.1.2. The Scene.....	10
2.1.3. The Viewer	10
2.1.4. Tri-relationships.....	11
2.2. 3-D Sensing.....	11
2.3. Structured Lighting	15
2.3.1. Is It A Good Approach?.....	15
2.3.2. Sparse vs. Dense : Assumptions about the Scene.....	18
2.3.3. The Correspondence Problem in Light Striping.....	19
2.3.4. Stripe Texture	23
2.3.5. Fusion of Light Striping and Intensity.....	24
2.4. Object Representation.....	25
Chapter 3 Surface Solution —	
Geometric Computation and Constraint Propagation	29
3.1. Oculomotor Cues and Calibration	30
3.1.1. Oculomotor Cues — Depth Information From Within The Eyes	30
3.1.2. Camera and Projector Calibration	31
3.2. Grid Point Computation in 3-D Space.....	36
3.3. Constraints	38

3.3.1. Basic Constraints	38
3.3.2. Geometric Rules	39
3.3.3. Topological Rules	42
3.4. Algorithms	46
3.4.1. Notations	46
3.4.2. Algorithm 1 — Stripe Extraction and Network Construction	47
3.4.3. Algorithm 2 — 3-D Computation for a Single Point	48
3.4.4. Algorithm 3 — Single Network Solution via Constraint Propagation ..	54
3.4.5. Algorithm 4 — Network Boundary Extraction	55
3.4.6. Algorithm 5 — Solution for the Scene	58
3.5. A Complete Example	60
3.6. Quantitative Analysis on the Degree of Ambiguity	68
3.6.1. Orthographic Projection.....	68
3.6.1.1. Assumptions.....	68
3.6.1.2. Basic Formulae	71
3.6.1.3. Computation in 3-Space.....	71
3.6.1.4. Computation in xy-Plane	73
3.6.2. Perspective Projection.....	75
3.6.2.1. Assumptions.....	75
3.6.2.2. Computation.....	76
3.7. Conclusions.....	78
Chapter 4 Inference of Surface Shape	79
4.1. Surface Fitting and Local Curvature.....	80
4.1.1. Differential Geometry — Local Surface Theory.....	80
4.1.2. Computation — Surface Fitting Using B-Spline.....	84
4.1.3. Examples.....	87
4.1.4. Occlusion Relationships between Virtual Edges.....	92
4.1.5. Summary	92
4.2. Surface Shape from Stripe Texture.....	94
4.2.1. Relations of Features in 2-D and in 3-D	94
4.2.2. Stripe Texture	97
4.2.3. Stripe Texture Classification.....	99

4.2.4. Experimental Results	101
4.3. Virtual edges	107
4.4. Discussion and Conclusions	109
Chapter 5 Multi-Channel Vision —	
Fusion of Light Striping and Intensity	111
5.1. Object Description	112
5.2. Light Striping is Not Enough.....	113
5.3. Comparison of the Two Information Channels	114
5.4. Stripe Extension.....	119
5.5. Interpretation of Contours.....	124
5.5.1. Apparent Curvature of Visual Contour and Local Surface Geometry ..	125
5.5.2. Combination of Image Contours and Stripes.....	127
5.5.3. Terminology and Definitions	128
5.5.4. Inference Rules	131
5.5.5. Some Mathematical Arguments about the Inference Rules	138
5.5.6. Implementation of Inference Rules	141
5.5.7. Examples.....	143
5.5.8. Experiments	147
5.6. Conclusions.....	148
Chapter 6 Concluding Discussion and Future Work	151
6.1. Summary	151
6.1.1. Computing 3-D Surface Solutions.....	153
6.1.2. Classification of Surface Patches.....	154
6.1.3. Fusion of Intensity and Stripe Image Data	155
6.2. Conclusion	156
6.3. Future Work.....	158
Appendix.....	159
Bibliography	170

List of Figures

Figure 1.1 - 3-D scene analysis scheme.....	3
Figure 1.2 - System diagram.....	7
Figure 2.1 - Depth cues.....	12
Figure 2.2 - Examples of structured light images.....	17
Figure 3.1 - Geometry of triangulation.....	31
Figure 3.2 - Sensing environment and coordinate systems.....	33
Figure 3.3 - Epipolar geometry.....	41
Figure 3.4 - Topological rule T1.....	43
Figure 3.5 - Two X-stripes cannot intersect in the image (Lemma 3.1).....	44
Figure 3.6 - Distinguishing X- and Y-stripes at an intersection point (Lemma 3.3)..	52
Figure 3.7 - X- and Y-stripes distinguished with respect to \vec{R}_i	53
Figure 3.8 - Algorithm 4 (Turning-Right algorithm).....	57
Figure 3.9 - A striped image of a block, a Coke can, and a piston.....	61
Figure 3.10 - Detected grid points in the jumble image.....	62
Figure 3.11 - Candidate labels for some grid points of the upper surface.....	63
Figure 3.12 - Valid and invalid combinations of surface solutions.....	66
Figure 3.13 - Ambiguity of location in 3-space.....	69
Figure 3.14 - Parallel projection model.....	70
Figure 4.1 - B-spline surface patch using 16 control points.....	86
Figure 4.2 - Histogram of curvatures of a planar surface patch.....	88
Figure 4.3 - Histogram of curvatures of a cylindrical surface patch.....	90
Figure 4.4 - Histogram of curvatures of an elliptical surface patch.....	91
Figure 4.5 - Light stripe networks in a striped image.....	95
Figure 4.6 - Stripe texture primitives.....	97
Figure 4.7 - Histogram on $\mu(\Delta_d)$ of stripe curves.....	103
Figure 4.8 - Histogram on μ_{cur} of stripe curves.....	104

Figure 5.1 - Fusion of intensity and light striping — example 1	116
Figure 5.2 - Fusion of intensity and light striping — example 2.....	117
Figure 5.3 - Striped image of an orange and the result of global thresholding	120
Figure 5.4 - Local thresholding results	121
Figure 5.5 - Segmentation using both stripes and intensity gradient.....	123
Figure 5.6 - Types of image contours	124
Figure 5.7 - Radial curvature and transverse curvature	126
Figure 5.8 - Intensity gradient image a sculpture, an orange and a block	127
Figure 5.9 - Four types of P-edge	130
Figure 5.10 - Inference rules 1-5.....	136
Figure 5.11 - Inference rules 6-13.....	137
Figure 5.12 - Lemma 5.1.....	139
Figure 5.13 - Lemma 5.2.....	140
Figure 5.14 - Fusion example 1 — image of a solid block.....	143
Figure 5.15 - Fusion example 2 — image of a sculpture, an orange and a block	145

List of Tables

Table 3-1 - Two Solutions for the Upper Surface of the Block	64
Table 3-2 - Summary of Surface Solutions.....	67
Table 3-3 - Number of Possible Locations in 3-D under Parallel Projection	75
Table 3-4 - Number of Possible Locations in 3-D under Perspective Projection	77
Table 4-1 - Surface Patches, Texture Primitives, and Classification	102
Table 4-2 - Average Gradient Values on 4 Features for 7 Surface Patches.....	106
Table 4-3 - Shadow and Shadow-Making Virtual Edges.....	108
Table 5-1 - Comparison of Two Information Channels.....	118
Table 5-2 - Summary of Labeling Results Using Inference Rules	148

CHAPTER 1

Introduction

The task of a general 3-D machine vision system is to recognize the three-dimensional objects in the field of view of the system's visual sensors, deduce the spatial relationships among the objects, and understand the configuration of the scene. This very complex task can be broken down into three basic subtasks : (1) extraction of useful information in 2-dimension and in 3-dimension, (2) construction of a description of the objects based on the information extracted, and (3) matching of the constructed description against the descriptions of the object models. These three subtasks are themselves complex and difficult enough that no general solution exists without various assumptions that considerably simplify the problem. The difficulties arise from the fact that the sensed images are merely two-dimensional projections of the three-dimensional objects, and therefore a great deal of information is lost. Furthermore, object models are themselves hard to build automatically, and there are infinitely many of them.

This thesis deals with problems in the first two subtasks, i.e. feature extraction and construction of object description (representation). In particular, a structured lighting approach will be presented that

- a) recovers the 3-D data of some surface points,

- b) deduces surface shapes using the 3-D data and the 2-D image features,
- c) develops a surface & boundary representation for later matching processes.

As with any other technique for machine vision, the approach presented in this thesis imposes certain assumptions on the environment of the vision system. But, these assumptions are quite *general*; they are not specially tuned for particular applications. Some basic assumptions are the following.

- (1) A controlled environment is assumed in which structured light is projected on the scene to create a striped image, and a diffuse light is used for obtaining an intensity image.
- (2) Multiple objects are likely to be in the workspace so that occlusions to both the projector and the camera are expected. Objects are assumed to have "arbitrary" but relatively smooth shapes. Objects are solid, opaque and static.
- (3) Imaging and projection use *general* positions.

Our experiments using structured lighting showed that this approach is promising not only in 3-D sensing, but also in feature extraction and shape inference. Relatively inexpensive equipment (a black-white camera, an image digitizer, and an ordinary 35mm projector) is sufficient to build the 3-D sensing system. This technique may find its value in many applications, especially in industrial environments such as automatic assembly control and robotics.

1.1 3-D Scene Analysis

A 3-D vision system is expected to recognize the objects in the scene and *where* they are. To accomplish this, it is often necessary to first obtain spatial information about the objects, the depth of some surface points, for example. The 3-D information obtained, as well as feature information extracted from the 2-D images, is then analyzed to draw inferences about the shapes of the objects. A proper description of the shape information

obtained is then developed in order to finally understand the scene. Surveys of 3-D scene analysis can be found in [Brady 82, Bajcsy 80], and it is outlined in Figure 1.1.

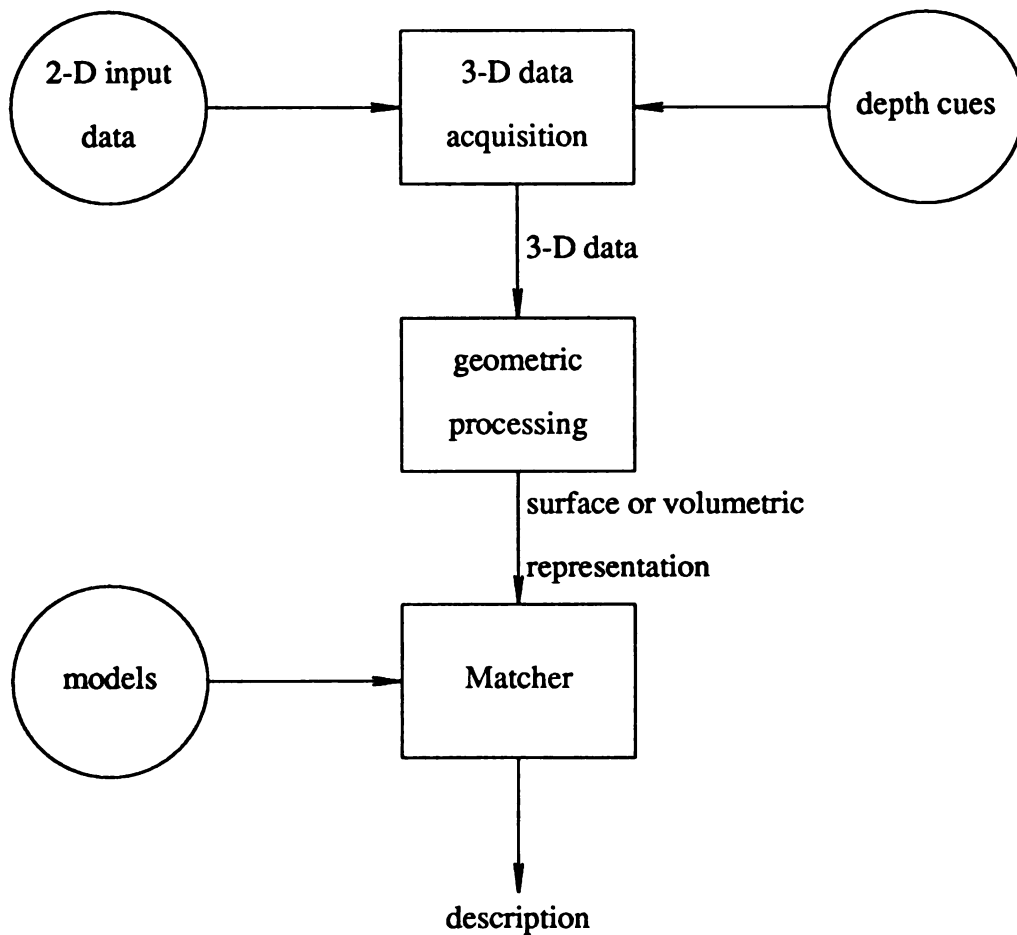


Figure 1.1 3-D scene analysis scheme

The first module in the diagram, *3-D Data acquisition*, is also called 3-D sensing, range sensing, or depth measurement. Many techniques have been developed, from time-of-flight direct sensing to stereo disparity measurement. Jarvis [Jarvis 83] and Strand [Strand 85] have provided overviews of various approaches in this topic. I will discuss these in more detail in later chapters.

Secondly, *object shapes* should be correctly inferred. Since there are so many different shapes, only a handful can be described in simple terms such as "planar", "cylindrical", "spherical", or "waved", etc. The majority are too complicated to describe in any easy way, such as a tree, a shirt, a telephone set, and the like. This difficulty leads to the problem of representation of the objects. A commonly used representation is to "describe" shape locally by *surface normals* in addition to position in 3-space. This representation gives surface orientation at every point of interest, and shows how the orientation changes from point to point. Marr and Nishihara called this type of representation the $2\frac{1}{2}$ D sketch [Marr and Nishihara 78]. Depending on how such an orientation map is obtained, approaches developed include *shape from texture* [Bajcsy and Lieberman 76, Kender 80], *from shading* [Horn 76], *from contour* [Stevens 79], *from stereo* [Marr and Poggio 76], *from motion* [Ullman 79], and so on. After surface shapes have been obtained, their relationships must be established in order to understand the scene. Objects may be partially occluded, in the shadow of others, touching each other, and at various distances. The relationships between surfaces (or surface patches), as well as the surfaces themselves and their associated edges should then be properly described using some representation, which is then fed to a model-matching module for recognition of individual objects and understanding of the entire scene.

There are other topics in 3-D scene analysis, but I shall discuss issues mainly in the two topics stated above, namely "3-D data acquisition" and "surface representation".

1.2 Problems Addressed in This Dissertation

In the Pattern Recognition and Image Processing (PRIP) lab of the Computer Science Department at Michigan State University, the components of a 3-D machine vision system have been developed consisting of 2-D processing for feature extraction, 3-D sensing, generation of 2½D representation, and model matching for recognition. It uses a light striping approach to get sparse 3-D data and a rough surface shape description, and refines it by using a gray-scale image of the same scene fused with the striped image. Integration of information from both striped image and gray-scale image is feasible in a controlled environment such as for automatic assembly and bin-picking for robotics. The overall system is depicted in Figure 1.2. The problems that I have been investigating and will be discussing in this thesis include the following.

- (1). *To obtain the 3-D surface data through the use of structured lighting, which projects a grid of light to illuminate the scene.* This process is a triangulating depth sensing with the camera-projector forming a stereo pair. The major difficulty, namely the line identification problem, is partially solved using a set of general constraints. Further breakdown of topics are :
 - camera and projector calibration
 - stripe extraction from the 2-D striped image
 - 3-D computation via triangulation
 - 3-D surface solution using general constraint propagation
- (2). *To deduce the rough surface shapes from the sparse 3-D data and from light stripe texture, which refers to the stripe patterns seen in the image.* This problem involves topics in surface fitting (B-spline), curvature computation (the Gaussian curvature and the two principle curvatures at surface points, as defined in classical differential geometry), and texture analysis (placement of the distorted stripe patterns in the image).

- (3). *To develop a surface representation suitable for later model matching.* The surface representation is a variation of Marr's 2½D sketch, including descriptions of edges, surfaces, and their relationships.
- (4). *To integrate information obtained from both the light striped image and the gray tone image of the same scene.* Multi-channel fusion helps to complement the two, and to reach a better description of the scene.

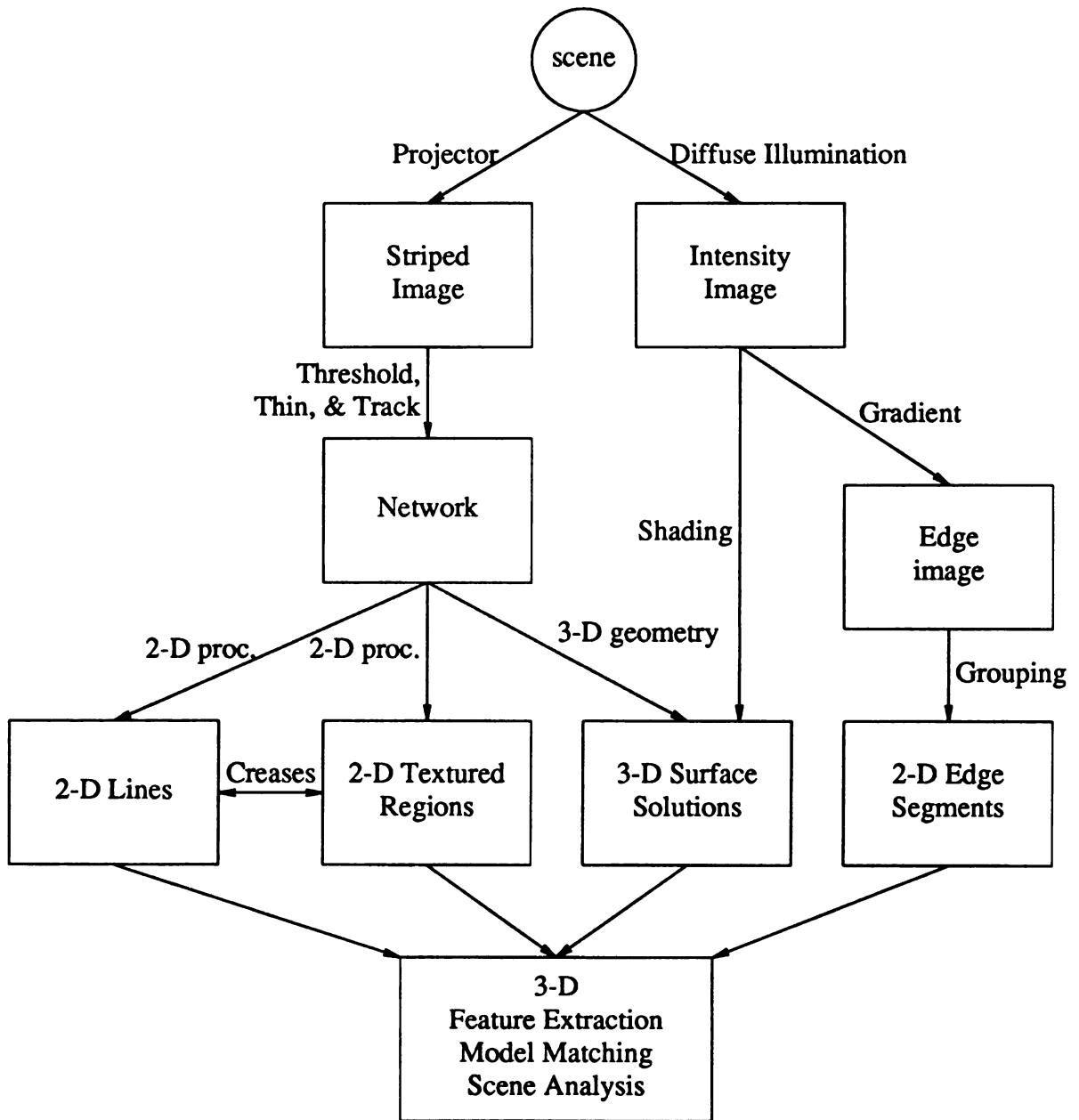


Figure 1.2. System diagram

CHAPTER 2

3-D Interpretation Using Light Striping

Structured lighting has been used in machine vision for about two decades, and now is regarded as a standard technique in machine vision. Most work using structured light has concentrated on 3-D sensing. In the MSU PRIP Lab, we have been using a structured lighting approach in 3-D surface sensing and in constructing surface representation as part of our 3-D vision system. Using light striping to create artificial light patterns on object surfaces, which are easy to detect and analyze, not only can we develop 3-D surface data at the selected surface points (on the light stripes), but also we can infer the shapes of surface patches directly from the light stripe texture seen in the 2-D image. This chapter will outline the three phases in surface representation using structured light: (1) 3-D sensing method using a grid of light, how the "line labeling" problem may be solved; (2) inferring surface shape via sensed 3-D data and stripe textures; and (3) constructing surface representation through combination of information from light striping and intensity. First, we briefly discuss the relationships between the light source, the scene and the viewer (camera).

2.1 Three Components of Visual Processing

A visual perception is the result of interaction between three participants : *the light source, the scene, and the viewer*. Each of the three is indispensable to a normal perception, and so are their relationships. The human visual system is capable of adjusting itself to adapt to the changing environment, so the three components of vision usually function in a nicely cooperative manner and result in a successful perception. But a machine viewer (machine vision system) may lack this capability. A proper arrangement of the other two components and their relationships is often necessary for the perception to be successful.

2.1.1 light source

In order to be able to see things, there must be at least a certain amount of light reflected (or originated) from an object to the retina. This constitutes the input end of the visual chain. We don't need to worry too much about the physics of the light itself, whether it's waves or quanta. We simply accept the laws of optics such as "light travels straight". But the physical properties of the illuminating light may considerably affect the visual perception, since change of the properties will alter the input to the processing chain. We want the light source to be neither too bright nor too dim and at a proper position such that the image sensed by the camera carries essential detectable features of the visible surfaces of objects in the scene, such as occluding boundaries, surface textures etc. Structured light, such as light stripes, is one such light source that makes surface features very salient and easy to detect. By properly adjusting the structured lighting equipment (a slide projector, for example), we can easily make the light stripes on surfaces to be the only detectable features, so that any other inherent surface textures or marks will not interfere with the imposed stripes.

2.1.2 the scene

Reflectance and other physical properties of the object surfaces play an important role in perception that enable us to discriminate one surface from the others through the *change* of the physical properties. For example, a piece of metal is distinguished from a piece of wood by their different surface reflectance that makes one darker or more textured than the other. It is this *contrast* that contributes to recognition. If there were no contrast, there would be no vision. But too slow or too rapid change of reflectance across the field of view (spatial frequency) prevents us from seeing the contrast [Campell and Robson 68]. In machine vision, the contrast of reflectance creates what are called "features" in an image. Feature extraction is one of the first modules in early vision; all further processing is based on whether the first module did a good job or not. Obviously, feature-less images provide no useful input to the processing chain. The objects in the scene must offer sufficient reflectance contrast at positions important to recognition. On the other hand, detectable contrast at irrelevant positions (noise) tends to destroy the structure of the object patterns and makes recognition harder. Images under structured lighting always provide surface features that are those imposed bright light stripes.

In short, the light source and the scene compose the source from which a good quality or a poor quality image might be produced. The success of the entire processing heavily depends on the quality of the image.

2.1.3 the viewer

Perception is the product of the viewer. Impairment of any part of the human visual system will greatly affect the final perception, regardless of whether the damage is in the eyeball, in the lateral geniculate nucleus, or in the visual cortex. A severe defect in any functional module in a computer vision system will lead to poor object recognition and hence poor understanding of the scene, no matter whether it occurs in the sensor, in

feature extraction, in representation, or in model matching. Because each module is studied and developed somewhat in isolation from the others, we often assume no significant defects (although there may be some small ones) inherited from preceding modules which produce input to the later modules.

2.1.4 tri-relationships

When looking at objects in front of the light source, we see a silhouette, whereas when the objects are lit from the front or side, their details look clearer. It is obvious that the relative position and orientation, or spatial relationships, of the light source, the objects, and the viewer, is critical to perception. Our aim is to infer their spatial relations, especially the surface orientation relative to the viewer from 2-dimensional images. This inference is a reverse procedure of imaging. Techniques that make such inference have been widely studied, designated by the name "shape from *x*", where *x* can be *shading*, *texture*, *contour*, *stereo*, *motion*, etc. Since the three components of vision interact in a very complicated way, we often want to set up certain conditions or assumptions that simplify the interaction of the three components. For example, the light source may be assumed to be a distant point source generating parallel illumination; the object surfaces may be assumed Lambertian so that light is reflected equally in all directions; the objects in the scene are usually assumed to be solid, opaque and relatively smooth. Under these conditions, various approaches to certain vision problems can be thoroughly studied.

2.2 3-D Sensing

It is remarkable that human beings are able to make distance judgements at all, let alone do so automatically and effortlessly. The retinal images from which depth information is extracted are two-dimensional and, therefore, inherently without depth. To extract information about the third dimension, the brain utilizes a variety of different depth cues,

providing a 3-D visual world. The sources of information about depth, or *cues* to depth, usually operate in harmony, yielding an unambiguous impression of 3-D space. Figure 2.1 shows different sources of depth cues [Gibson 50].

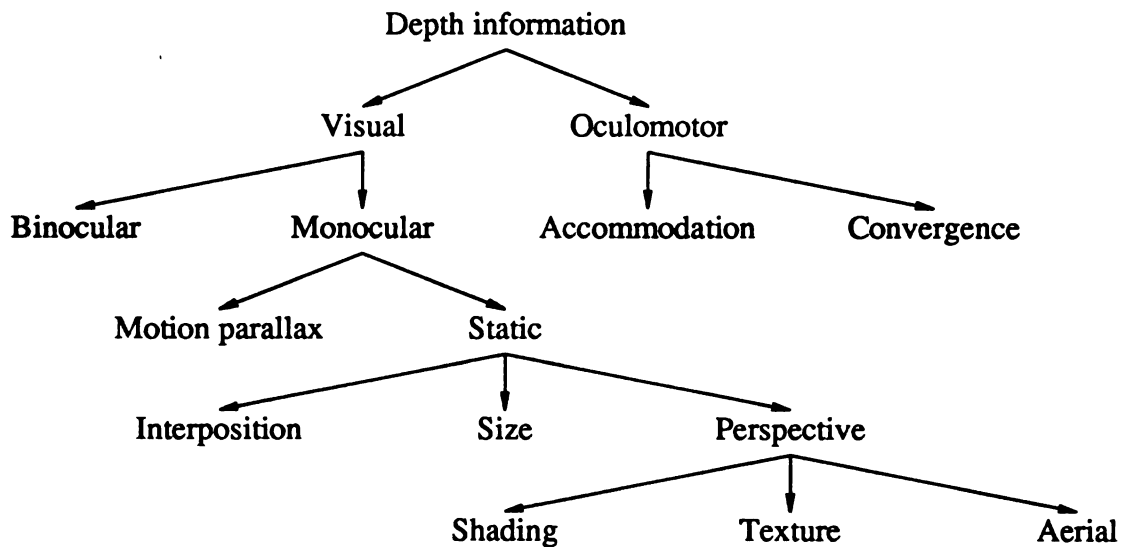


Figure 2.1 Depth cues

In these categories, oculomotor cues (accommodation of the lens and angle of convergence of two eyes) are the only ones that provide unambiguous information about absolute depth, the distance from an observer to an object. In contrast, any of the visual cues can provide good information about relative depth, the distance between different objects or different parts of a single object. But they must be supplemented with other information in order to specify absolute distance.

Analogously, for a computer vision system, the camera image consists of a complex distribution of intensity, contour and color, contained in a flat, 2-dimensional picture. To provide 3-D data from the 2-dimensional picture is just like solving for three unknowns using only two equations. We must somehow find a "third equation". The "third equation" must be *independent* of the two already provided, i.e. supplementary information must be obtained from somewhere other than the input picture itself, if absolute depth is to be recovered. As examples, the supplementary information may be obtained from additional images taken at different positions in space or in time, from a sensor calibration process, from knowledge of the specific application domain, or from natural constraints in general. Information from different sources or channels is expected to be consistent and yield good surface solutions. This principle of integrating information from multiple sources is not only valid for 3-D sensing, but also important in later higher-level processing.

3-D sensing methods can be divided into several categories, according to how the additional information is obtained.

- (1). **Direct Sensing (Time-Of-Flight).** The distance of an object is determined by measuring the time it takes for light to travel from the source to the object and back (receiver is at the same location as the source). In practice, the difference in phase between the transmitted and received signals is measured rather than the time of flight. The phase shift is directly (but not uniquely) related to distance. A representative of this approach is a laser range finder, which can sense the distance to any surface point in its field of view. This approach measures distance with no image processing. It doesn't really establish 3-D data for surface points, rather it provides only information about depth. In this sense, this method should be called *third-dimension sensing* instead of 3-D sensing.

- (2). **Shape From x .** As mentioned earlier, x may be texture, shading, contour etc. Methods in this category develop 3-D surface shape in terms of local surface normals, i.e. local surface orientation with respect to the camera, provided that the camera viewing direction (optical axis) is known. They do not really provide absolute distances to objects but only relative distances or orientations. It is good enough in situations where only object shapes are important, but does need extra information to determine the absolute distance for, say, robot motion control purposes. In situations where absolute distances are not critical to our tasks and we are only interested in surface shapes or orientations, shape-from- x may be a good approach to achieve our goal.
- (3). **Stereo.** Emulating human binocular vision, the stereo approach to 3-D sensing employs two cameras that have a known relative displacement and a known convergence angle of their optical axes. By measuring the disparity between the two images, depth can be calculated via triangulation. As has been emphasized in the literature, the major difficulty is the "correspondence problem" — given an image feature in one image, to identify the image feature in the other image such that the two are created from the same physical surface feature.
- (4). **Structured Lighting.** In illuminating the scene, natural ambient light is replaced by an artificial light source, which can be of any structure (pattern) that is convenient for the task. Using structured light by itself is not an independent approach to 3-D sensing; the underlying means is still a monocular (shape-from) or binocular (stereo) method, but under different illumination conditions and hence facing re-phrased problems. If a single light beam or a single light plane is used as the light source, the underlying method is direct sensing through triangulation. If a uniform grid of light is the light source, the underlying method is stereo and analysis of textures and contours. The major advantage of using structured light over ambient light is that features in the images are better defined. Image features are

easier to detect; their relationships are more regular (parallel, equi-spaced, etc.) following the property of the generating pattern of the light source, they prominently reveal the surface geometry that humans can readily use to interpret the scene.

2.3 Structured Lighting

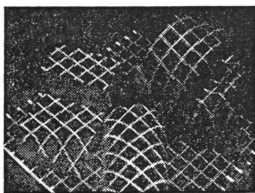
Because of the potential advantages, we use the structured light approach not only for 3-D sensing, but also for surface shape inference from 2-D stripe textures. In this section, I shall discuss some issues in using structured light: stripe labeling (the correspondence problem), comparison with the stereo approach, previous work using structured light, and how light striping can help to infer surface shapes.

2.3.1 Is it a good approach?

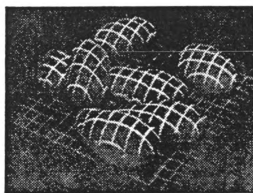
Often, the illumination is ambient, using diffused light. Every point (up to the resolution granularity of the viewer's sensor) in the field of view that is illuminated is seen by the viewer, and provides a piece of information about that point. The total input to the processing system consists of millions of such pieces of information that are to be resolved. For a machine vision system, how to pick useful ones from the millions is often a very difficult task. Edge detection, for example, is commonly considered as the first thing to do in the process, that finds *features* that are the significant changes of input information from point to point. Because of the inevitable noise introduced under ambient illumination, even the best edge detection methods followed or preceded by a noise removal process may never work well on a complex noisy image. If the first processing step is deficient, how could we expect the later steps down the processing chain to perform well when they are based on the output of the first processing module? One way to solve this problem is to get "good" input in the first place, which provides just as much useful information as necessary for the recognition purpose, and as little

information as possible that may interfere. Structured lighting is one of the alternatives that may achieve this goal.

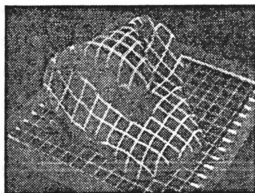
Structured lighting is an *active* approach that projects onto the scene a pattern that has a geometrically regular distribution which is known *a priori*. A laser beam generating device or a slide projector can be used to generate the structured light patterns. The most often used structured light patterns include a single light ray that generates a single bright point in the scene, a single light sheet that produces a bright line or curve on the object surfaces, parallel light sheets that cause parallel bright lines or curves, or a grid of light that casts a grid pattern on object surfaces. Figure 2.2 shows some example images taken under illumination of a grid of light.



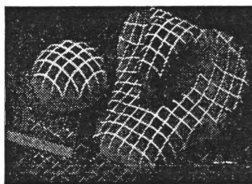
(a)



(b)



(c)



(d)

Figure 2.2 Example of structured light images

Structured light is good for the following reasons. First, it changes the image of features by imposing artificial features on the object surfaces. These artificial features are very easy to detect. Images thus obtained are directly "enhanced" by the structured light rather than by applying image enhancement algorithms. Second, false features caused by noise under ambient illumination are greatly reduced in the sense that the noise would not be present in a light striped image where only bright light lines or grids are visible to the camera and elsewhere is assumed totally dark. Thus a binary image with no false parts due to noise is easily created. Third, unlike passive methods, a structured lighting system processes only a small portion of the 2-D image (only bright pixels on light stripes are of interest), resulting in considerable savings of storage and processing time. Last but not least, the distortion of the projected grid pattern is a function of surface shape; the 2-D light stripe networks alone convey a great deal of information about object shapes, which would not be directly available otherwise. In addition, an exactly registered intensity image of the same scene is easy to obtain by simply turning off the projector and turning on another light which may be placed at the position of the camera: hence the intensity image can be fused with the striped image to provide combined information that would not be available in either of the two alone.

2.3.2 Sparse vs. dense : assumptions about the scene

The structured lighting approach deals with *sparse* surface points and hence gains storage and time savings in computation. The trade-off is that the information about many other surface points (not on bright stripes) is lost. What are the effects of this loss on our goal of object recognition? In other words, we want the sampled surface points to be dense enough to correctly represent the surface, and sparse enough to keep as little superfluous information as possible.

How densely the surface points should be sampled depends on the spatial characteristics of the surface and the granularity we want to describe the surface. For example, if we knew a surface is an ideal planar surface, it would suffice that only three non-linear points be computed in order to determine the plane, and a few boundary points for describing the surface. Any information more than that is superfluous. On the other hand, if the surface undulates in a complicated fashion, we need to collect information from many more points or we will obtain a seriously defective surface description. The problem is that we do not know *a priori* what kind of surfaces we are about to handle. On the contrary, to determine the surface characteristics is our goal. Hence, the density of sampled points, or equivalently the stripe spacing in structured lighting environments, can only be determined under certain assumptions about the object surfaces. In other words, the sparse light striping method would work, under these assumptions:

- (1) Surfaces are *smooth* and are of "low order" in the sense that the spatial frequency of surfaces is less than the stripe frequency; i.e. there is small spatial change on the surfaces between consecutive stripes.
- (2) Surfaces are "much" larger than stripe spacing so that a surface patch is covered by a multi-stripe network.

Thus it is *assumed* that light stripes capture the essential features of the surfaces and that the features missing from striping are insignificant. The class of objects this approach may work on includes objects of smooth surfaces that are not too small relative to the stripe spacing. This class of objects is often encountered in industrial applications, such as bin-picking, where the possible objects are known *a priori*.

2.3.3 The correspondence problem in light striping

Binocular visual information : stereopsis

In humans, the two eyes look at much the same region of the visual space, only near the margins of the visual field do the two eyes provide exclusive monocular coverage. You may ask, "Why did nature position our eyes in such a way as to provide two views of the world, thereby duplicating a significant portion of these views?" The answer may be that the slight differences, or *disparities*, between the view seen by the left eye and the view seen by the right eye enable us to make exceedingly fine depth judgements that are simply impossible when using just one eye. Psychologists have shown that we can tell a 1 millimeter difference from a distance of 1 meter, which corresponds to a judgement accuracy of one-tenth of 1 percent. In other words, the resulting disparity between the two eyes' views is less than four ten-thousands of a millimeter, many times smaller than the diameter of a single visual receptor in your eye [Sekuler and Blake 85]! Because of this extraordinary resolving power, stereopsis has been extensively studied throughout the literature.

The correspondence problem

In order to use stereo as a depth sensing model in machine vision, we have to first solve the correspondence problem, i.e. to identify the two matching points, one in each image, between which the disparity occurs. The difficulties of the correspondence problem arise due to the following reasons. (1) There must be at least some feature points (edge points, say) in the images; images of homogeneous properties make the matching impossible. (2) The overlapping portion of the field of view of the two images must be such that a feature should appear in both images; so features in one image but missing from the other will considerably complicate the matching process. And (3), the matching process is computationally prohibitive due to combinatorial explosion if the number of feature points is large. Julesz [Julesz 71] developed random-dot stereogram, both halves of which consist of nothing more than a random array of black and white dots. The two halves are identical except in one of the two halves of the stereogram a central subset of dots has been shifted laterally by several rows. This lateral displacement creates a retinal

disparity between the two halves, and yields patterns separated in depth when viewed through a stereoscope. Since each of the two halves is made up of numerous random dots, it is hard to imagine how the "feature matching" is undertaken because there are so many tiny dots in one image that would match up with any single dot in the other image.

The grid line labeling problem in light striping

A structured lighting system for depth sensing is essentially a stereo system, in which the camera forms one view and the structured light generating device (a slide projector, say) is considered as the second view. The only difference is that "imaging" direction at the second view is reversed. As far as the geometry in 3-D computation is concerned, imaging or reverse imaging makes no difference. Pretend the light is "reflected" from the object surfaces to the projector. The projector would "see" a regular structured pattern on its "image plane" which is exactly the pattern on the slide. Hence the camera image and the projector "image" compose a stereo pair. Since the features of interest are the light stripes, the feature matching between the two images is to tell which grid line in the projector image generates a particular stripe in the camera image. Thus, the correspondence problem becomes the *grid line labeling*, or *grid line identification* problem. Notice that in the camera-projector stereo the grid lines are complete in one image of the stereo pair (the projector image), hence a matching will always succeed although some stripes may be missing in the camera image due to occlusion.

Let's now discuss how the grid lines might be identified. First, if only one single line is projected, there is one stripe in the image. This stripe may be broken into pieces because of occlusion or discontinuity of object surfaces, but all these pieces correspond to the only projected line. The line identification problem is immediately solved. But, the projected line needs to be swept across the scene in order to obtain the depth measurements at various points that cover the entire scene. Scanning usually is slow and requires a special device which ought to insure the accuracy of the calibrated setup during the

sweeping. Shiri and Suwa used this simple stripe lighting scheme for polyhedron recognition [Shiri and Suwa 71]. The range finder they developed employed a rotating slit projector which projects a light beam on the objects. While the projector is rotated, pictures are sampled by a TV camera at predetermined times and the information related to each picture and the corresponding slit beam angle is stored. The 3-D position of each slit is calculated via trigonometry. Agin and Binford [Agin and Binford 71], instead of using a rotating projector, employed a rotating mirror, which reflect the light beam onto the scene. The 3-D positions on that stripe were calculated. The mirror then rotated a small angle and the procedure repeated. After a sweep was finished, the orientation of the light lines was rotated by 90° , and a second mirror scan sweep was taken. This resulted in data for an overlapping grid of laser lines covering the scene.

In cases where multiple lines are projected at once, it would not be trivial to distinguish one line from the others provided that all lines are identical. If each of the projected lines bears a unique property, such as color [Yang, Boyer and Kak 84], thickness [Le Moigne and Waxman], spatial code [Posdamer and Altschuler 82], etc., the problem will be easier to attack. But the desired unique properties may themselves be hard to extract from the image and sometimes require special equipment to generate (color, for example). In using spatial code, identical lines are projected, each of which is assigned a unique spatial code in advance. A sequence of images are taken, each with certain projected lines on or off according to the spatial codes assigned to each line. For example, four projected lines are assigned codes 1,2,3 and 4, i.e. 001, 010, 011, 100 in binary. Three successive images are needed to encode the lines. If a line is seen only in the first image it is line number 1; if it is seen in the first and the second images, it must be the line number 3, and so on. If N lines are projected, the spatial code length is $\log_2(N)+1$. That is, $\log_2(N)+1$ images are required. This method makes line labeling simpler, but requires multiple frames and more importantly requires very precise alignment of these frames.

It appears desirable to use a "snapshot" scheme — one single image of multiple lines, which requires neither special devices nor multiple images. The trade-off is that the line identification will become more complicated. If the projected lines can be identified, their 3-D locations can be calculated via triangulation. Our approach for identifying the projected lines is to find a set of labels a line may possibly be assigned, compute the 3-D locations for each candidate label and get rid of those that violate certain constraints. The constraints are very general ones that are available in the real world. A few or hopefully a unique line label may survive the constraint tests. The line labeling method using constraints for solving for 3-D surface points will be discussed in detail in Chapter 3.

2.3.4 Stripe texture

Using projection of a grid of light, not only can we obtain data of surface points in 3-space, but also the stripe patterns (or the texture of the stripe networks) cast on the surfaces provide clues about surface shape. From the striped image examples in Figure 2.2, we perceive 3-D shapes through the 2-D stripe texture alone. This is because the distortion of the stripe pattern from its generating pattern (a regular grid) directly relates to the surface shape.

Research shows that surface orientation (slant and tilt) can be resolved from surface texture, texture gradient in particular [Gibson 50]. Shape-from-texture methods have been used to recover shapes [Bajcsy and Lieberman 76, Ikeuchi 84, Kender 78] for many years. In the case of light striping, the texture elements (texels) are those "quadrilaterals" externally imposed on the object surfaces in the form of a light grid, rather than an intrinsic property of the surface as most natural textures are. The good thing about stripe texture is that it is more *regular* and more *structured* than most natural textures, and easier to extract and analyze. Moreover, because the stripes are the intersections of 3-D planes (planes of light) and the object surfaces, they reveal surface shapes in a more

deterministic manner since the geometry of the 3-D planes are known in advance. Assume that the relative displacement of the projector and the camera is fixed as it is in our experiments, the ambiguity of the relation between a 2-D image stripe and its corresponding 3-D surface stripe is very much constrained by the fact that the 3-D surface stripe is also a planar curve (in the plane of a projected light sheet) that may not freely flap in the 3-space to image in the same observed 2-D stripe. This property does not hold for natural textures. In Chapter 4, we shall discuss the issues of how the surface shapes can be deduced from 2-D stripe textures.

2.3.5 Fusion of light striping and intensity

Although light striping is a promising approach in machine vision, it may not be useful for probing information about surfaces that have no stripes due to occlusion of the projector (shadows). Also, the stripe networks may not give an accurate segmentation of the image, due to the stripe fading effect on curved objects. To overcome these weaknesses, among others, we might need information from some sources other than light striping. An intensity image of the same scene provides such a source, on which classical image analysis such as edge detection can be applied. Edges are regarded as primary information for object recognition, but they are subject to a high percentage of errors (noise), provide little information about surfaces, and need a significant amount of time to process.† Combination of information from the two sources — intensity and light striping — is needed. In short, light striping provides *surface* information, while intensity provides *boundary* information. The fusion of these two provides a *surface-boundary* representation of the objects in the scene, which will be the subject of Chapter 5.

†Actually, human ability at determining shape from line drawings — even corrupted ones — is uncanny [Biederman 87]. General shape inference by machine using corrupted boundary data remains an unsolved problem.

2.4 3-D object representations

Object recognition is done by comparing or *matching* features extracted from the image with precompiled object models. To carry out the matching, the observable (image features) and the target (model) must be described with the same representation scheme. The choice of a representation depends not only on the object domain, but also on the matching scheme. There are three categories of representation on the basis of their dimensionality of spatial description. These three categories are 2-D, $2\frac{1}{2}$ -D, and 3-D representations. The details of these representations are surveyed in [Chin and Dyer 86] and [Bhanu and Ho 87].

In 2-D representation, objects are described using a set of one or more distinct views. Each view is treated independently. The 3-D problem is reduced to 2-D using 2-D image features and their relations as primitives [Lieberman 79]. A commonly used approach is to organize global features of an object into a *feature vector*, or *geometric property list*. The global features include area, perimeter, compactness, elongatedness, center of gravity, moments of inertia, number of holes, number of corners, etc. Selection of features can be determined by training [Gleason and Agin 79]. Matching schemes using feature vector models usually involve statistical pattern recognition techniques, such as the Bayes classifier, nearest neighbor classifier, and decision-tree classifier [Agin and Duda 75]. In the feature-vector approach, if features are easy to compute, object representation is compact, and the matching is fast. But this approach requires a separate model for each possible 2-D view of an object; and it is unable to handle occlusion. Researchers have also developed various other methods using 2-D representations, such as *line and arc boundary segments* model [Perkins 78], *Fourier descriptors* [Persoon and Fu 77], *hierarchical feature* model [Shirai 78], *local evidence accumulation via clustering* [Stockman et al 82], and *template matching* in generalized Hough transform space [Ballard 81], among others.

In $2\frac{1}{2}$ -D representation, features are defined in "surface space" with respect to the view coordinate system. Local surface properties such as orientation, depth, discontinuities, reflectance, etc., as well as 2-D features (points, lines), are used in the description of objects. Representations of this kind are the *intrinsic images* [Barrow and Tenenbaum 78], the $2\frac{1}{2}$ -D *sketch* [Marr 78], the *needle map* [Horn 79], *parameter map* [Ballard 81], and *surface-orientation-map* [Brady 82]. In industrial parts-recognition applications, the use of range maps and local surface-orientation maps are of particular interest. A great deal of effort has been made in recent years in segmentation of range data into surface patches [Bolles 81, Bolles and Fischler 81, Henderson 82, Henderson and Bhanu 82, Milgram and Bjorklund 80, Sugihara 79, Besl and R. Jain 85, Hoffman and A. Jain 87]. Many shape-from techniques that aim to derive the surface-orientation map have been proposed [Horn 75, Bajcsy and Lieberman 76, Stevens 81, Witkin 81, Woodham 78]. Using $2\frac{1}{2}$ -D geometric features to represent rigid objects, matching can be done via discovery of a linear transformation (translation, rotation and scaling) of the iconic object model that put all salient features into correspondence. The sensed data are considered an instance of the model through the transformation. The transformation is found through clustering in the "transformation space" [Chen and Stockman 86, Stockman 87]. This method can handle multiple "general" objects and partial occlusion; it allows imperfect sensed data, and features can be added or deleted during the process. But for multiple object scenes, especially for objects of arbitrary shape, the computational cost may be rather high. *Interpretation tree search* [Grimson and Lozano-Perez 84, Chen 86] is also a feasible approach to matching. This backtracking approach keeps track of the mutually consistent pairs of sensed feature and model feature along the path searched in the tree. There are many ways to incorporate heuristics to speed up the search process, for example, by consideration of features in some special order or by using quick constraint checking. Another popular approach using $2\frac{1}{2}$ -D representation is the relational-feature

graph [Oshima and Shirai 83, Fan and Medioni 87, Hoffman and A. Jain 87]. In a relational-feature graph, each node represents a surface patch with labels (planar, convex, concave, ellipsoid, cone, cylinder, etc.), and an edge represents the relationship between the two adjacent surface patches. Each edge is associated with a set of properties, such as angle between the two surfaces, relative positions of the centroids, etc. Matching is performed by comparing an observed graph with a set of graphs, one for each viewpoint of each object modeled.

3-D representation is a viewpoint-independent representation using an object-centered coordinate system. It describes objects in "object space". Objects are represented using surface patches [Baumgart 72, Shneier 81], spines and sweeping rules, volume descriptors such as *generalized cylinders* and *generalized cones* [Binford 71, Brroks 83, Nevatia and Binford 77], or multi-view features [Koenderink and vanDoorn 79, Goad 83]. 3-D models allow the most general and complete descriptions of objects, but require the most extensive and sophisticated processing.

In our work, we construct a $2\frac{1}{2}$ -D representation using the features extracted from the striped image and the gray-tone image. The representation is in the $2\frac{1}{2}$ -D category since the features are obtained from a single viewpoint and they contain geometric surface properties such as 3-D location, orientation (normals) and shapes, as well as the 2-D features (edges). Because each surface patch is defined associated with its boundary edges, we also call the representation a *wing* representation. In particular, the representation consists of the following basic elements.

1. surfaces

- (a) qualitative — type (planar, convex, concave, cylindrical, etc.),
- (b) quantitative — boundary, 3-D location, orientation;

2. edges

- (a) qualitative — type (extremum, blade, fold, mark, shadow),
- (b) quantitative — location, length

3. surface-surface relationships

- (a) occlusion,
- (b) adjacency,
- (c) belonging to same or different objects;

4. surface-edge relationships

- (a) surface boundaries,
- (b) surface marks,
- (c) shadow edges,
- (d) intersection (common border) of two surfaces.

This representation is a surface-boundary representation similar in purpose to Marr's $2\frac{1}{2}$ D sketch [Marr 82]. Upon construction of the image representation, image features that correspond to object shapes are made explicit — they can be described *symbolically*. This representation is to be matched against object models that are constructed independently. We are currently investigating modeling and matching processes based on the *wing* representation [Stockman and Chen 87, Stockman et al 87]. The ultimate goal is to recognize objects and understand the scene.

CHAPTER 3

Surface Solution — Geometric Computation and Constraint Propagation

It is often desirable for a machine vision system to collect 3-D surface data as the first processing step (3-D sensing), because many geometric inferences about the object surfaces can be made through the 3-D surface data. 3-D surface data may be expressed in terms of some coordinate system or expressed as ranges (distances from the surface points to the viewer). As discussed in Chapter 2, 3-D sensing techniques are divided into four categories — direct sensing, shape-from-x, stereo, and structured light. This chapter will describe a method to compute surface data using light striping, in which the *stripe identification* problem is partially solved using general constraints. These constraints are based on the assumption that objects in the scene are solid, opaque and static. Once the stripes are identified, the surface solution is calculated through triangulation. Experimental results show that this approach often produces surface solutions with a small degree of ambiguity and sometimes a unique solution is reached. In case of multiple solutions, any of the locations in 3-space can be used to infer the surface shape.

3.1 Oculomotor Cues and Calibration

This section deals with the camera calibration process. In order to obtain depth information using triangulation, the parameters of the sensors, such as the focal length and the direction of the optical axis, should be fixed and known; or equivalently, the 3-D to 2-D projection should be fixed and known. In human vision, these parameters are regarded as oculomotor cues; whereas in machine vision, they are obtained through a calibration process.

3.1.1 Oculomotor cues — depth information from within the eyes

Among the various cues to depth perception, as briefed in section 2.2, the oculomotor cues are the *real* ones that relate the information directly to the distance from you to the objects [Wallach and Floor 71, Sekuler and Blake 85]. Oculomotor cues refer to **accommodation** of the eye lens and the **angle of convergence** of the two eyes. The former is a measure of how much the lens is strained when your eyes fixate on an object; while the latter provides information on the directions of the optical axes of the two eyes. Suppose we were able to attach tiny gauges to the two sets of muscles that control the lens and the eyes' inward-outward movement. The readings on the gauges, or the degree of muscular contraction, are transferred to somewhere in the brain that can figure out the two values — angle of convergence and the amount of accommodation. The distance would then be computed by solving the "trigonometry equation" that relates distance to the two values. One amazing thing about this is that the whole process is done in milliseconds and the depth judgement is made *continuously* as the fixation point moves from one place to another. But both accommodation and convergence operate over a limited range of distance. When you focus on an object from a distance of 20 feet and beyond, the muscle controlling accommodation assumes its most relaxed state; at the same time, the convergence angle vanishes to zero. So the oculomotor cues are relied on only within a region of space immediately in front of you, otherwise the resulting distance measure

won't be reliable.

The geometric principle of the distance computation is illustrated in Figure 3.1. If the displacement of the two viewing points (two eyes) and the two optical axes are known, the triangle can be easily solved. It is obvious that if the two optical axes are almost parallel, or equivalently the two angles θ_1 and θ_2 are close to 90° , a very small inaccuracy in the measurements will introduce a large error in the computed distance.

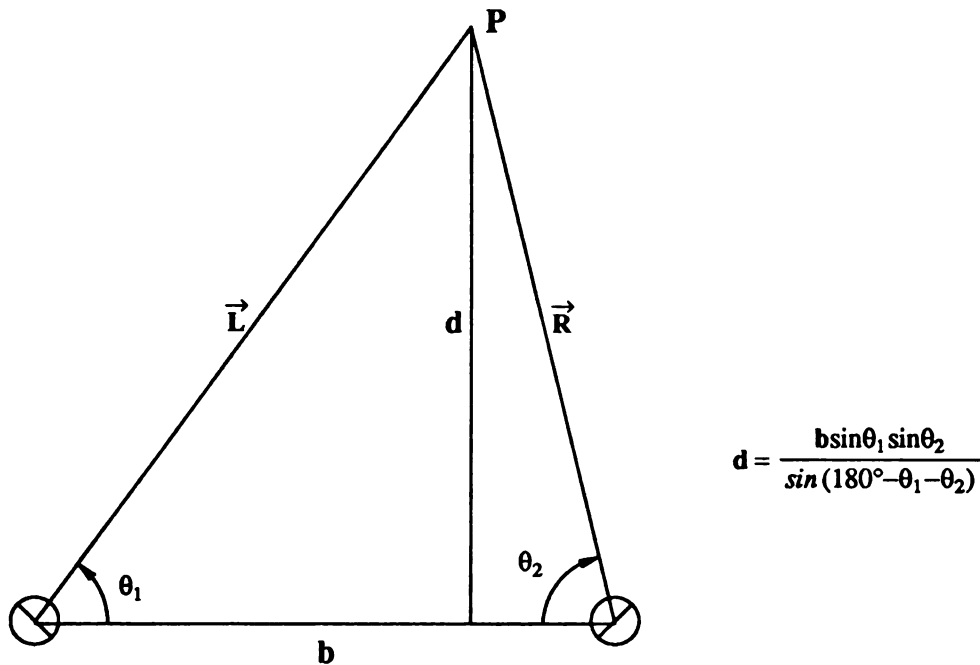


Figure 3.1 Geometry of triangulation

3.1.2 Camera and projector calibration

In order for triangulation to be applicable to depth computation in a system with a camera-projector setup, we must have some "oculomotor" information available. The procedure that gains this information is known as *calibration*. A calibration procedure

estimates internal parameters of the camera such as focal length, direction of the optical axis etc., embedded in a transformation that geometrically relates a point in 3-space and its projection point in the image plane. The transformation is often expressed as a 3×4 matrix that transforms points in one coordinate system into points in another coordinate system, with rotation, translation and scaling [Faig 75, Hall et al 82, Yakimovsky and Cunningham 78, Tsai 86]. We first describe our coordinate systems.

Sensing environment and coordinate systems

Figure 3.2 is a sketch of the sensing environment used to create images such as the one in Figure 2.2. Three coordinate systems are defined in this environment: the global or world coordinate system, the camera coordinate system, and the projector coordinate system. The global world coordinate system is fixed on the worktable with its xy plane coincident with the table plane and z axis pointing up; all output surface and edge solutions are in terms of the global coordinates. The camera coordinate system has origin at the center of the camera lens with the z axis towards the focal point. The front image plane is at $-f$ on the z axis, where f is the focal length of the camera lens. The image is digitized at 512×512 resolution, while the 2-D digitized picture counts its rows (x axis) from top to bottom and columns (y axis) from left to right. The projector coordinate system is similar to the camera coordinate system, except that the unit is *grid line number* rather than *pixel*. $M \times N$ grid lines are used (in the experiments, we used M, N in the range from 14 to 21). The spacing of the lines is about 15mm on the worktable and the working field-of-view is about 250mm square.

We also define two terms *up* and *down* as direction indicators in the image plane. The optical axis of the projector is considered directional from the focal point to the worktable. The projection of the optical axis in the image plane defines the up-to-down direction in the image. Some times we also use the terms *above* and *below*. These terms will be used in Chapters 4 and 5.

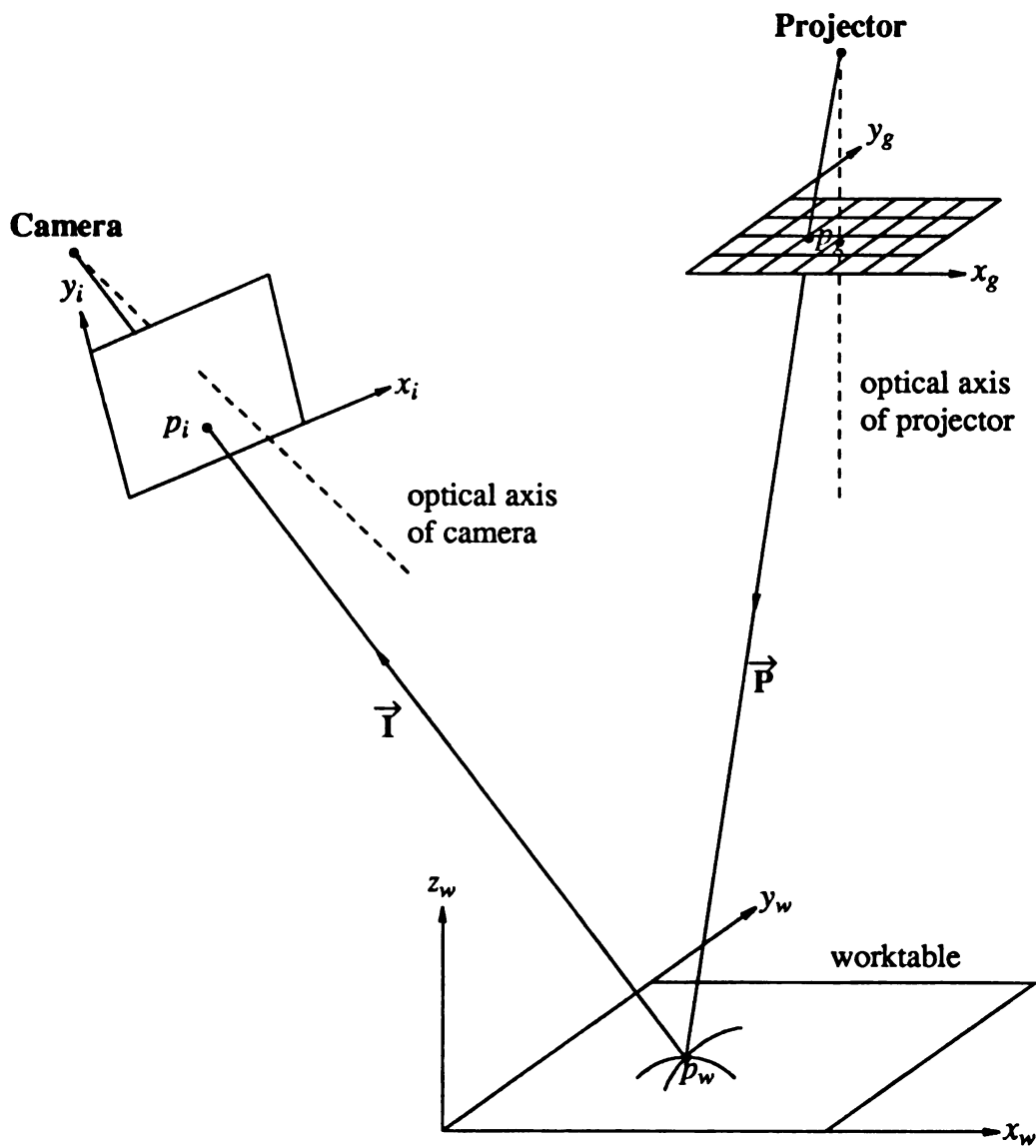


Figure 3.2 Sensing environment and coordinate systems

3-D to 2-D transformation

We use (x_w, y_w, z_w) , (x_i, y_i, z_i) and (x_g, y_g, z_g) to denote a world point in 3-space, image point in camera coordinate system, and grid point in projector coordinate system, respectively. The subscripts w , i , and g stand for world, image, and grid. Since the image plane is fixed at $z_i = \text{constant}$, all image points have the same z coordinate, we need only (x_i, y_i) to represent an image point in the image plane. Similarly, we use (x_g, y_g) for a grid point in the projector plane.

The transformation from one coordinate system to another is commonly defined as a homogeneous 4×4 matrix \mathbf{M} . Given a point (X, Y, Z) in one coordinate system, its coordinates (x, y, z) in the transformed coordinate system are

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Since in our case the image plane and the projector plane have constant z values in their coordinate systems, the transformation from the world coordinate system to the camera coordinate system would have the form

$$\mathbf{M}_C \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} tx_i \\ ty_i \\ t \\ 1 \end{bmatrix} \quad (3.1)$$

and the transformation from world to projector has the form

$$\mathbf{M}_P \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} sx_g \\ sy_g \\ s \\ 1 \end{bmatrix} \quad (3.2)$$

where t and s are scaling factors. We also call \mathbf{M}_C the camera calibration matrix, or camera matrix for short; \mathbf{M}_P the projector calibration matrix or projector matrix.

Calibration

Calibration is to determine the homogeneous transformation matrices M_C and M_P , each of which has $3 \times 4 = 12$ entries. Because one of the 12 entries can be made unit (matrix scaling), there are 11 entries to be estimated for each matrix. Let's take the camera matrix as an example. Let

$$M_C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix},$$

equation (1) becomes

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} tx_i \\ ty_i \\ t \end{bmatrix}.$$

That is,

$$x_w c_{11} + y_w c_{12} + z_w c_{13} + c_{14} = tx_i \quad (3.3)$$

$$x_w c_{21} + y_w c_{22} + z_w c_{23} + c_{24} = ty_i \quad (3.4)$$

$$x_w c_{31} + y_w c_{32} + z_w c_{33} + 1 = t \quad (3.5)$$

Substituting for t in (3.3) and (3.4) by the left hand side of (3.5), we get two linear equations

$$x_w c_{11} + y_w c_{12} + z_w c_{13} + c_{14} - x_i x_w c_{31} - x_i y_w c_{32} - x_i z_w c_{33} = x_i \quad (3.6)$$

$$x_w c_{21} + y_w c_{22} + z_w c_{23} + c_{24} - y_i x_w c_{31} - y_i y_w c_{32} - y_i z_w c_{33} = y_i \quad (3.7)$$

where the 11 c_{ij} 's are unknown. To solve for the 11 unknowns we need at least 11 equations. Since each 3-D point (x_w, y_w, z_w) and its corresponding image point (x_i, y_i) offer two

equations, we need measurements of no less than 6 points which give 12 equations. 25 points were used in experiments reported here to overdetermine the 11 c_{ij} 's using least-squares solution. Usually 20 to 30 may be sufficient to guarantee that enough "good" points are included and are widely scattered covering the field of view of the camera.

The same procedure applies for projector calibration. Let $M_P = (p_{ij})_{3 \times 4}$ be the projector matrix, we have two equations

$$x_w p_{11} + y_w p_{12} + z_w p_{13} + p_{14} - x_g x_w p_{31} - x_g y_w p_{32} - x_g z_w p_{33} = x_g \quad (3.8)$$

$$x_w p_{21} + y_w p_{22} + z_w p_{23} + p_{24} - y_g x_w p_{31} - y_g y_w p_{32} - y_g z_w p_{33} = y_g \quad (3.9)$$

By measuring enough light stripe intersection points in 3-D and knowing the grid labels x_g, y_g , we solve for the 11 p_{ij} 's using a least squares method. The procedure described was adapted from [Ballard and Brown 85].

More information about calibration can be found in [Tsai 86], where many calibration techniques are surveyed and a high precision method is proposed.

3.2 Grid Point Computation in 3-D Space

After calibration is done, M_C and M_P are fixed. Our next task is to determine locations in 3-space of points that are observed in the image. So we rewrite (3.6) to (3.9) as

$$(c_{11} - x_i c_{31})x_w + (c_{12} - x_i c_{32})y_w + (c_{13} - x_i c_{33})z_w = x_i \quad (3.10)$$

$$(c_{21} - y_i c_{31})x_w + (c_{22} - y_i c_{32})y_w + (c_{23} - y_i c_{33})z_w = y_i \quad (3.11)$$

$$(p_{11} - x_g p_{31})x_w + (p_{12} - x_g p_{32})y_w + (p_{13} - x_g p_{33})z_w = x_g \quad (3.12)$$

$$(p_{21} - y_g p_{31})x_w + (p_{22} - y_g p_{32})y_w + (p_{23} - y_g p_{33})z_w = y_g \quad (3.13)$$

with x_w, y_w, z_w as unknowns (See [Hall et al 82]).

The calibration procedure established a model under which the geometry of the projection process and the imaging process are fixed through equations (3.1) and (3.2), or equivalently (3.10) to (3.13). From the equations we see that given a 3-D point $p_w = (x_w, y_w, z_w)$, its corresponding points $p_i = (x_i, y_i)$ in the image plane and $p_g = (x_g, y_g)$ in the projector plane are well defined. On the other hand, given an image point p_i , there exist infinitely many 3-D points that may be imaged at the same point p_i . They are those that lie on the same imaging ray. Mathematically speaking, for a given $p_i = (x_i, y_i)$ there are two equations, namely (3.10) and (3.11), for 3 unknowns x_w, y_w, z_w . The solutions would be infinitely many. Geometrically, each of the equations (3.10) and (3.11) defines a plane in 3-space; the intersection of the two planes is a 3-D line, or imaging ray, that defines the solution space. Any point on that line satisfies (3.10) and (3.11). By the same token, a projector ray emitting from a given grid point p_g defines an array of points in 3-space (eq. (3.12) and (3.13)).

Now let's turn to the "triangulation" computation for (x_w, y_w, z_w) . Suppose we have identified the correspondence between an image point p_i and a grid point p_g , that is, we have determined that p_i is the image of a 3-D point (unknown (x_w, y_w, z_w)) that was created by projecting p_g onto the scene. Each of the imaging and the projection point will bring us two equations; we end up with 4 equations in 3 unknowns. The 3-D point (x_w, y_w, z_w) can now be determined. Theoretically, it is the intersection of the imaging ray and the projection ray. In practice, because of inaccuracy in calibration and discrete imaging, we will not get two rays intersecting in 3-space. The solution would then be the point in 3-space that has least distance from both rays.

We return to the assumption that the correspondence has been established. But how? When we see a bright spot in the image, its coordinates in the image plane (x_i, y_i) are known. But from which grid point (all grid points look alike) in the projector slide is this image point created? If the grid point (x_g, y_g) is unknown, the above computation wouldn't work — 4 equations with 5 unknowns!

Here is where the constraints come in.

3.3 Constraints

The problem we are facing now is to determine which of the grid points in the projector plane is the one that creates the image point we are looking at. Once the *right* one is determined, (3.10) - (3.13) are good for solving for (x_w, y_w, z_w) . The first step is to discard those grid points that can not possibly match the image point (x_i, y_i) , using a set of general constraints that are available in the real world. The grid points that survive the constraint test are considered "candidates" for matching the image point.

3.3.1 Basic constraints

First, we assume that the scene to be analyzed is from the real world, and that the objects in the scene are solid, static, opaque, and that their surfaces undulate slowly at the scale of the grid spacing.

Two constraints on physical world objects were identified by Marr and Poggio [Marr and Poggio 76] in their stereo disparity computation : (C_1) a given point on a physical surface has a unique position in space at any time instant, and (C_2) the surfaces of objects are generally smooth compared to their distances from the viewer and matter, divided into objects, is cohesive. They are called the uniqueness constraint and the continuity constraint.

Since each object has some geometric form that occupies a certain volume and objects do not overlap in space, not only is a single surface point uniquely determined in space, but also a cohesive set of object points is uniquely determined in space. Stated in other words, if a given space volume is occupied by one object, it cannot be occupied by any other object at the same time. Thus (C_1) is extended to a volumetric uniqueness constraint: a given solid object has a unique volume in 3-D space at any time instant. Under the assumption of objects being opaque, a rule corresponding to (C_1) applicable in

surface computation may be stated as follows: two physical objects have non-overlapping 2-D images in any viewing (projecting) direction. This rule is very useful in reducing ambiguity of surface solutions.

The continuity constraint (C_2) asserts that if a continuous curve segment is present in the 2-D image and is assumed to lie on one object surface in 3-D space, then points on that curve must have continuous 3-D positions. Thus, neighboring points on a surface constrain each other, or in Marr and Poggio's terms, they *inhibit* or *excite* among themselves depending on their computed 3-D coordinates. This rule restricts possible grid line label assignment to neighboring points, and hence is also useful in removing ambiguity of surface solutions.

We now refine and materialize the uniqueness constraint and the continuity constraint into a set of geometric and topological rules useful in surface computation using structured light. The algorithms based on these rules are thus very general since these rules are from the real world with no oversimplification or application-specific assumptions being made other than the general assumptions already stated.

3.3.2 Geometric rules

From the uniqueness constraint C_1 , which says that a given solid object has a unique volume in 3-D space at any time instant, the first three geometric rules follow. We add general position and work volume rules.

- (G1) (Epipolar line — limit of degree of ambiguity) *For each pixel in the image there exists a small set of projector rays that could possibly image there.*
- (G2) (Uniqueness of single points) *Any image point may be the projection of at most one projector ray.*
- (G3) (Uniqueness of solid objects) *Two stripe networks in an image correspond to non-overlapping regions in the projector slide plane.*

(G4) **(General position)** *If two stripe endpoints lie on the same projected ray in 3-D then their stripe numbers are the same and one lies on a surface between the other and the projector.*

(G5) **(Work volume)** *The (X,Y,Z) coordinates of any world point must lie within the field-of-view of both the camera and the projector, and be above the support plane ($Z = 0$).*

These geometric rules are explained as follows.

For (G1), because a surface point (a stripe intersect, say) has a unique position in 3-D space, the projector ray that creates that bright spot and the camera ray that images the point are also uniquely determined in 3-D space. The chance for other projector rays to intersect the same camera ray in space, in order to image at the same pixel, is slim. Although in practice we do not require the rays to intersect exactly, the number of projected rays that are *close* to the camera ray is still small, provided a general viewing angle. The size of the set of those projected rays determines the degree of ambiguity of 3-D solutions for the surface point and depends on the number of stripes available, stripe spacing, camera distance from the scene, and the error model. The ambiguity analysis will be discussed in section 3.6.

Let's now find *what* are these possible projector rays. Consider a true but unknown 3-D point p_w that is projected from a grid point p_g in the projector plane, and imaged at p_i in the image plane. Figure 3.3 illustrates the situation. The straight line connecting p_w (or p_i) and the camera focal point is the imaging ray \vec{I} ; the straight line connecting p_w (or p_g) and the projector focal point is the projecting ray \vec{P} . Because the two rays are not colinear in general and they meet at p_w , they determine a plane that intersects the projector plane at a line \vec{e} , which is called the **epipolar line** of the imaging ray \vec{I} [Faugeras 86].

Since all 3-D points along the imaging ray are candidates for the unknown p_w , each of which is sourced from a point on the epipolar line \vec{e} , all possible grid points must be

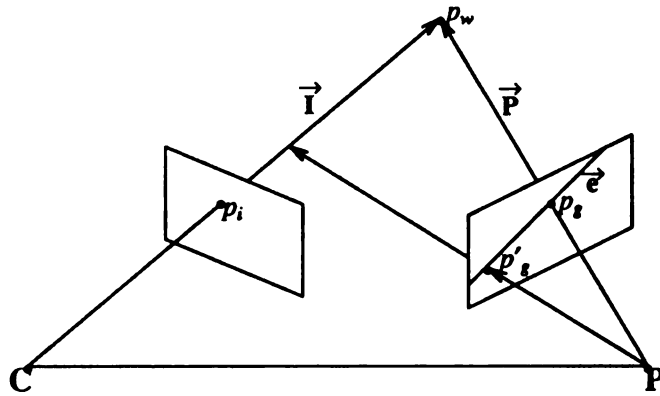


Figure 3.3 Epipolar geometry. Point p_i in the left image may correspond only to points on the epipolar line \vec{e} in the right image.

sought on \vec{e} . Given an image point, its epipolar line can be computed by computing the imaging ray using (3.10) and (3.11), projecting two points (not too close to each other) on the imaging ray onto the projector plane using (3.12) and (3.13), then forming the epipolar line using the two projected points. The epipolar line runs across the grid lines on the projector slide, with intersections being those candidate grid points. A detailed algorithm will be given in section 3.4.

(G2) and (G3) are directly from the uniqueness constraint (C1). By (G2), a grid point in an image may be assigned a single stripe number at a time. (G3) is useful for checking if two computed 3-D surface patches are valid or not. Their positions in 3-space must be such that no overlap should occur if they are projected in either the projector-projection direction or the imaging direction.

Rule (G4) applies to occluding-occluded relationships. Because objects have unique volumes in 3-D, their 2-D projections on the projector slide plane do not overlap. In cases where the two regions in the slide plane touch, a point on the touching boundary corresponds to two 3-D points lying on two 3-D surfaces that differ in depth, one

occluding the other.

(G5) is obvious.

3.3.3 Topological rules

The continuity constraint C_2 says that object surfaces are generally smooth (compared to their distances from the viewer) and points on object surfaces have continuous positions in three-space. No matter what kind of objects are in the scene, the 3-D stripes on the object surfaces preserve their topological relationships. This allows the following topological constraints :

(T1) *If a grid intersect is assigned grid-line number pair (x_g, y_g) , the only possibilities for its neighbors in the connected 2-D stripe network, at most four, are (x_g-1, y_g) , (x_g+1, y_g) , (x_g, y_g-1) and (x_g, y_g+1) .*

(T2) *In 3-D, X stripe-curves (Y stripe-curves) do not intersect other X stripe-curves (Y stripe-curves).*

(T3) *Any two stripe curves can intersect at most once in the image.*

(T4) *A continuous (smooth) 2-D curve indicates a continuous (smooth) 3-D curve.*

(T5) *A continuous (smooth) 2-D net indicates a continuous (smooth) 3-D surface.*

These topological rules are further described below.

For (T1), because the grid lines on the slide are ordered, from 1,2,...,to M in x direction, from 1,2,...to N in y direction, their projection on object surfaces (3-D stripes) are ordered exactly the same, no matter what. These stripes are imaged in the image plane with their order preserved also, as long as no viewing accident occurs. See Figure 3.4.

T2 claims that no X stripes can cross in 3-space. Do we have the same rule in the 2-D image? The answer is yes, provided that the stripes have zero thickness (mathematical curves) and surfaces are smooth. This is shown in the following lemma.

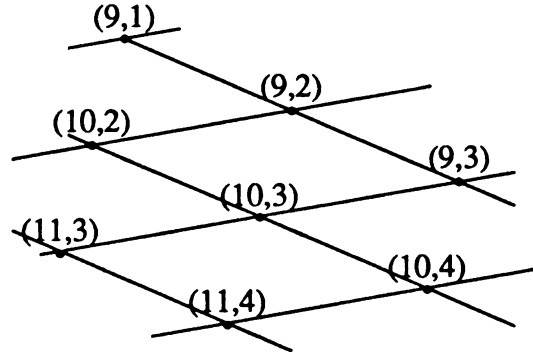


Figure 3.4 Topological rule T1. Labels of neighboring grid points in the image must differ by one unit of either x_i or y_i .

Lemma 3.1. Let S be a surface, $p \in S$ be a point on S . Let their projection in the image be S_i and p_i . If S is smooth at p , then there is at most one X-stripe in the image that passes through p_i .

proof. We prove that if an X-stripe in the image X_{g_i} passes through p_i , no other X-stripes can pass through p_i .

Referring to Figure 3.5, let the focal point of the camera be C , \vec{v}_p be a vector on the line of sight (pC) in the direction from p to C , \vec{n}_p be the surface normal at p .

Since p is visible, $\vec{n}_p \cdot \vec{v}_p > 0$. Because S is smooth at p , the surface normal is continuous at p . So, there exists a region $\delta_p \in S$ such that for all points $q \in \delta_p$, $\vec{n}_q \cdot \vec{v}_q > 0$; i.e., the entire surface patch δ_p is visible to the camera. Hence, there is a “general cone” determined by (C, δ_p) , which is divided by δ_p into two parts : the visible part (from C to δ_p) and the invisible part (occluded by δ_p).

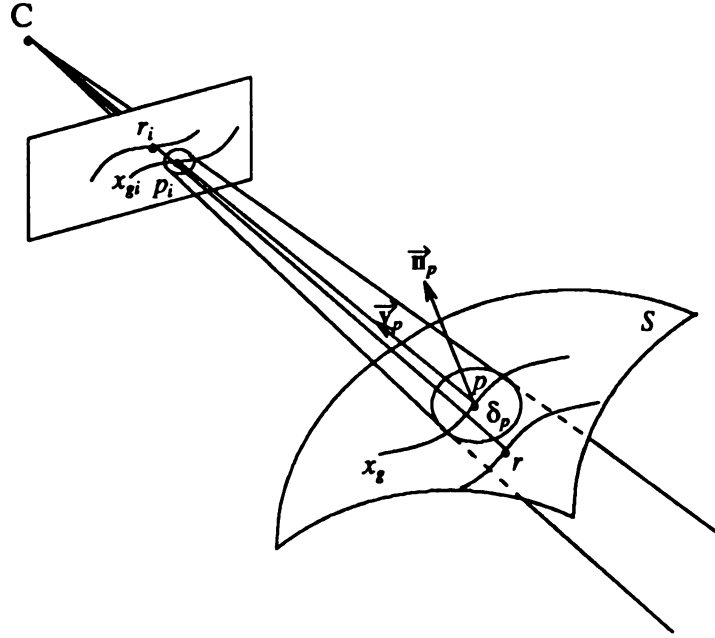


Figure 3.5 Two X-stripes cannot intersect in the image (Lemma 3.1)

Since stripe spacing d is a constant (> 0), δ_p can be chosen such that its “diameter” is smaller than d . In other words, any point r on other X-stripes in 3-D space may not fall in the surface patch δ_p . Thus, for any point r , there are 3 cases :

- 1). $\vec{n}_p \cdot \vec{v}_r \leq 0$, r is invisible;
- 2). $\vec{n}_p \cdot \vec{v}_r > 0$, and r falls inside the occluded volume in the cone, it is invisible;
- 3). $\vec{n}_p \cdot \vec{v}_r > 0$, and r falls outside the occluded volume in the cone; it is visible but its projection r_i in the image plane is outside the image of δ_p , and hence may not be coincident with p_i .

So, as long as p is visible, any point on any other X-stripe cannot pass through p_i in the image. Since p is an arbitrary point on an X-stripe, no two X-stripes can intersect in

the image. ■

Note : In practice, since images are discretely quantized, stripes have non-zero thickness (at least 1 pixel wide), and the image of the small region δ_p may not be smaller than one pixel, two X-stripes may meet at a common image point in this discrete case, but this occurs infrequently.

T3 is an extension of T2. It is proved in the Lemma 3.2.

Lemma 3.2. Let S be a smooth surface. Any two stripes on S can intersect at most once in the image.

proof. By Lemma 3.1, we only need to prove that an X-stripe X_{gi} and a Y-stripe Y_{gi} in the image intersect at most once. Note that the subscript "i" implies image plane, and no subscript implies 3-space. Assume they intersect at p_i . For any point $q_i \in Y_{gi}$ and $r_i \in X_{gi}$, where $q_i \neq p_i$ and they are projections of 3-D points $q \in Y_g$ and $r \in X_g$, there is a distance ("spacing" > 0) between q and r . Since S is smooth at r (r may or may not be the same point p), for X-stripe X_g visible at r and by the same argument in the proof of Lemma 3.1, q_i , which is the projection of q in the image, must be outside a δ_{ri} area in the image. Hence Y_{gi} cannot intersect X_{gi} at q_i which is different from p_i . ■

In practice, due to blooming effects, surface discontinuities, and viewing angle, T2 and T3 may not hold. But the situations where T2 and T3 break down are infrequent. Thus we still consider T2 and T3 to be valid in 2-D with a footnote that says "most of the time". By the same argument, the topological rules T4 and T5 can be established.

These constraints are similar to some of those of Lowe and Binford [Lowe and Binford 85]

3.4 Algorithms

After the geometry for 3-D computation and constraints are discussed, we are now ready to develop the algorithms to find surface solutions. These algorithms, given striped image input, will produce surface locations in 3-space with as little ambiguity as possible. Before these algorithms are applied, the camera and the projector are calibrated; the two calibration matrices are known. Notice that the calibration process needs to be done only once for successive images, as long as the camera-projector setup remains fixed.

3.4.1 Notations and Data Structure

As we have already been using, we shall use subscript "i" to indicate items in the image plane, "g" for items in the grid plane (projector plane), and "w" for items in 3-D world, or no subscript at all in cases where no confusion would occur.

image-point : $p_i = (id, x_i, y_i, degree)$, where id is unique to each p_i , (x_i, y_i) are image coordinates, and $degree$ is the number of its neighbors in the image network.

grid-point : $p_g = (x_g, y_g)$, the coordinates in the projector plane.

3D-point : $p_w = (x_w, y_w, z_w)$, the 3-D coordinates of p_w .

igw-point : $p = (p_i, p_g, p_w)$, a 3D-point p_w created by grid-point p_g , and seen in the image as image-point p_i .

network : $n = (V, E)$, where $V = \{p_{ij}, j=1, \dots, |V|\}$ is a set of image-points, and $E = \{(p_{ij}, p_{ik}), p_{ij}, p_{ik} \in V\}$ is the "neighboring" relation on V . Consider the network n as a planar graph, V is the set of vertices, E is the set of edges.

stripe-curve : $g = 3\text{-tuple } (stripe\#, XorY, P)$, where $stripe\#$ is the id of the stripe, $XorY$ indicates this stripe being an X-stripe or a Y-stripe, P is an ordered set of igw-points, ordered on x_g or y_g depending on the value $XorY$.

surface : $s = 3\text{-tuple } (id, type, W)$, where $type$ is a classification of the surface (planar, convex, concave, spherical, etc), W is an ordered set of stripe-

curves sorted on *stripe#*, X-stripes first followed by Y-stripes.

scene : set of surfaces (and later, their relationships).

3.4.2 Algorithm 1 — stripe extraction and network construction

The purpose is to segment the 2-D image into a number of visible surfaces as indicated by connected components of stripes. We call the connected components of stripes *networks*.

input : A striped image.

output : Set of 2-D networks, each is a set of connected stripes in the image.

method : We only give a brief outline, omitting many technical details.

- (1) Threshold the striped image to create a binary image in which pixels on bright stripes are one and other pixels are zero.
- (2) Apply a thinning algorithm to make the stripes one pixel wide. Clean up small spurs caused by thinning.
- (3) For each bright pixel, compute its *degree* (of value 1, 2, 3, or 4) by counting the number of bright pixels in its 3×3 neighborhood. Pixels of degree 1 are stripe end-points. Pixels of degree 3 or 4 are stripe intersection-points. Pixels of degree 2 are connecting points on stripes. Since the thinning operation often generates degree-3 points (as well as degree-4 points) at the crossing of two stripes, a simple clustering algorithm is applied on the detected degree-3 and degree-4 points with cluster radius of about 5 to 8 pixels depending on the width of the stripes before thinning. The cluster centers are taken as the crossing points.
- (4) Extract a network n_j by tracking the stripe points. $V_j \leftarrow \{\text{end-points and intersection-points tracked}\}$. $E_j \leftarrow \{(p_{ik}, p_{il}), p_{ik}, p_{il} \in V_j \text{ and connected by pixels of degree 2}\}$. $n_j \leftarrow \{V_j, E_j\}$. Reset all stripe points tracked to zero.

- (5) Repeat (4) until all pixels in the image are zero.

Since steps (1), (2) and part of (3) are performed by hardware (in our case on a VICOM image processor), the time complexity is mainly due to steps (4) and (5). The clustering step in (3) examines only the neighborhood of degree-3 and degree-4 points; it takes time linear in the number of these points. For (4) and (5), stripe points occupy only a small fraction of the whole image, and each stripe point is examined once. Hence the time complexity is linear in the number of stripe pixels.

3.4.3 Algorithm 2 — 3-D computation for a single point

The purpose of this algorithm is to find for each imaged grid point the set of possible surface points that may have created it.

input : Calibration matrices M_C (camera) and M_P (projector), and an image-point $p_i = (id, x_i, y_i, \circ)$.

output : Set of 3D-points each of which satisfies the projection equations (2.1) and (2.2). In other words, this algorithm finds 3D-points possibly created by grid-points and imaged at p_i . In addition, XY-stripe-ness of p_i and its neighboring image-points are determined.

method : (1) Obtain the camera ray \vec{R}_C (a 3-D line) using M_C and (x_i, y_i) . This takes $O(1)$ time.

(2) If \circ of $p_i \neq 1$ (an intersection point) then

(i) Form epipolar line : Project the camera ray \vec{R}_C onto the projector plane using M_P to get the epipolar line \vec{e} of \vec{R}_C . Time cost is $O(1)$. See Figure 3.3 for the epipolar geometry.

(ii) Grid line intersections that are closest to \vec{e} are selected (one for each X-grid-line). For each selected $p_s = (x_s, y_s)$, form a projector ray \vec{R}_P , and compute the distance d between \vec{R}_C and \vec{R}_P . If $d < \text{threshold}$, \vec{R}_C and \vec{R}_P

are considered to intersect in 3-space and $p_g=(x_g, y_g)$ is one candidate for creating the image point p_i according to the geometric constraint G1; the corresponding 3D-point p_w is calculated and igw-point $p=(p_i, p_g, p_w)$ is included in the output. Since each X- or Y-stripe is examined once, it take $O(\max(N_X, N_Y))$ time where N_X and N_Y are the numbers of X-grid-lines and Y-grid-lines, respectively.

- (iii) Once a grid point (x_g, y_g) is obtained in (ii), the image \vec{R}_i , of the projector ray \vec{R}_p , can be computed using matrix M_C . The XY-stripe-ness of p_i and its neighbors are determined with respect to \vec{R}_i as described in Lemma 3.3 below. This part of computation needs $O(1)$ time.

Else (a stripe end point)

compute 3D-point p_w , the intersection of the camera ray \vec{R}_C and each X-light-plane $(x_g, 0)$ or Y-light-plane $(0, y_g)$, depending on whether the image-point p_i lies on an X-stripe or a Y-stripe, and then output $p=(p_i, p_g, p_w)$. Time cost $O(\max(N_X, N_Y))$,

The time complexity of this algorithm is $O(\max(N_X, N_Y))$: linear in the number of grid lines.

Lemma 3.3. X-stripes and Y-stripes in an image can be distinguished.

proof. Referring to Figure 3.6 we define the following notation.

- P_X : light plane that creates stripe X_g in 3-D,
- P_Y : light plane that creates stripe Y_g in 3-D,
- P_i : image plane,
- \vec{R} : intersection of P_X and P_Y , a projector ray that produces the intersection point A of stripes X_g and Y_g ,

- C : focal point of camera,
 X_{gi}, Y_{gi} : projections of stripe X_g and Y_g in image plane P_i ,
 A_i : image of A in P_i ,
 \vec{R}_i : image of \vec{R} in P_i , be used to indicate the up-down sense in the image.
 P_e : epipolar plane of \vec{R} , a plane determined by point C and line \vec{R} . Notice that \vec{R}_i is the intersection of the image plane P_i and the epipolar plane P_e .

Under the "general viewing position" assumption, P_e is not coincident with either P_X or P_Y . Because the stripe X_g as a space curve $z = f(x, y)$ is a single-valued function along the curve, and P_e divides P_X into two half-planes, it is clear that X_g is also divided into two half-stripes lying at different sides of P_e . At the same time P_e also divides the image plane P_i such that the right half-plane of P_X is mapped to the right half-plane of P_i and the left half-plane of P_X is mapped to the left half-plane of P_i . So, the two half-stripes of X_g are projected to different sides of \vec{R}_i in the image plane P_i ; that is to say, X_{gi} must cross \vec{R}_i once. A similar statement holds for Y_{gi} .

Take two points v_i and w_i on the right half-plane of P_i (at the right side of \vec{R}_i) such that $v_i \in Y_{gi}$, $w_i \in X_{gi}$ and they have same image x-coordinate. Let their counterparts in 3-D be $v \in Y_g$ and $w \in X_g$. Because X_g and Y_g lie on object surfaces and objects are assumed opaque, points "below" the stripes in the P_X and P_Y planes are invisible. Referring to Figure 3.6, let the line Cv intersect the light plane P_X at v' . For both v and w visible, v' must lie on the visible ("above") side of X_g in P_X . Thus the projection of v in the image P_i must be "above" that of w .

Because the pair (v_i, w_i) was arbitrarily chosen, every point on Y_{gi} must be "above" its corresponding point (with same image x-coordinate) on X_{gi} , if there is one. So, the entire half-stripe Y_{gi} (on the right side of \vec{R}_i) is "above" the half-stripe X_{gi} . The same reasoning can be applied to the left half-plane of P_i , except that "above" is changed to "below".

We thus conclude that if an X-stripe and a Y-stripe intersect in the image, they can be distinguished with respect to \vec{R}_i , the image of the projector ray that produces the intersection point. ■

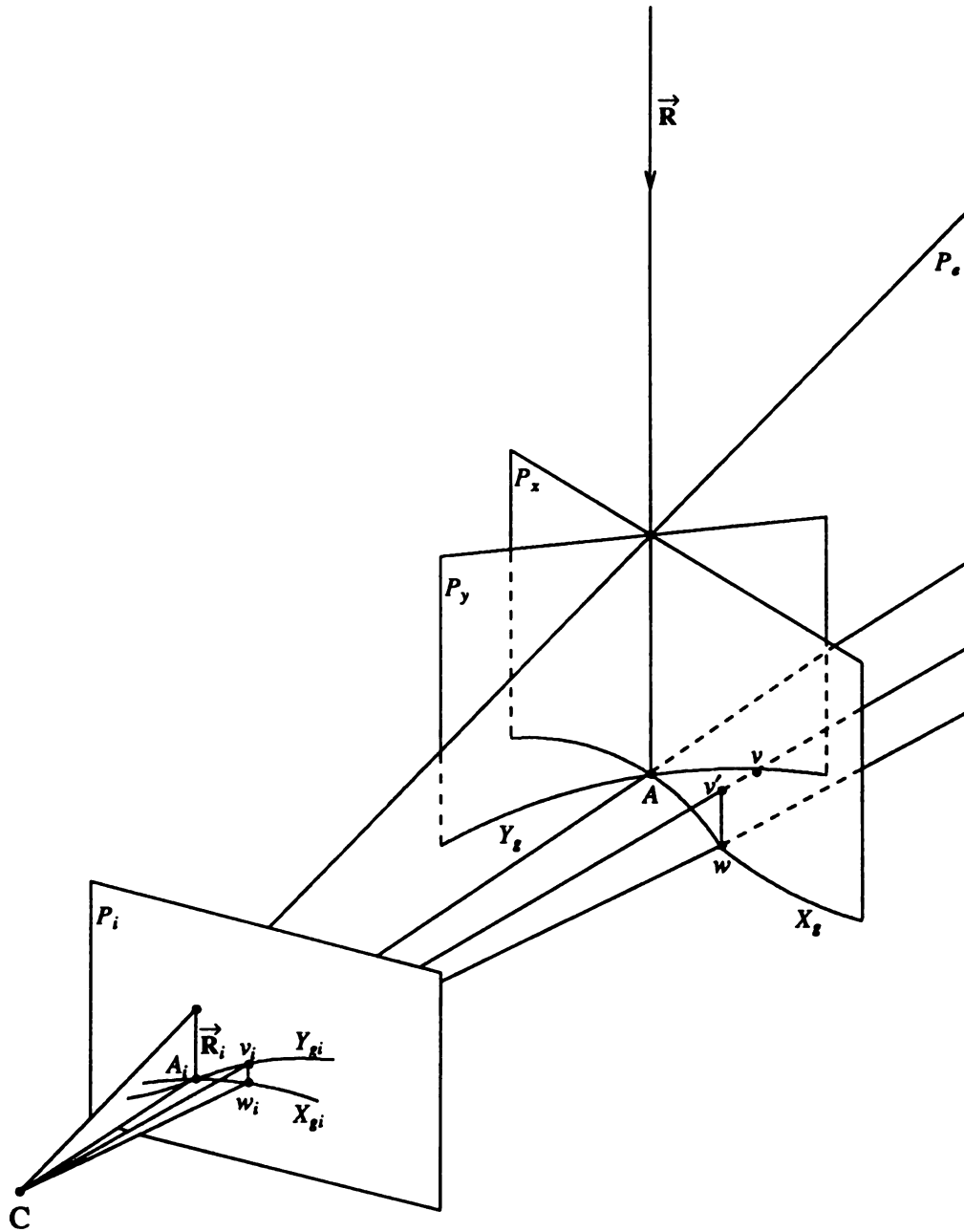


Figure 3.6 Distinguishing X- and Y-strips at an intersection point (Lemma 3.3)

The procedure of making that decision at an intersection point A_i is the following :

- (a) Project the projector ray \vec{R} onto the image plane P_i , obtain \vec{R}_i .
- (b) From a point $s \neq A_i$ on \vec{R}_i , moving clockwise, we will encounter stripes of Y-X-Y-X (or of X-Y-X-Y, depending on the camera position), alternately. See Figure 3.7.

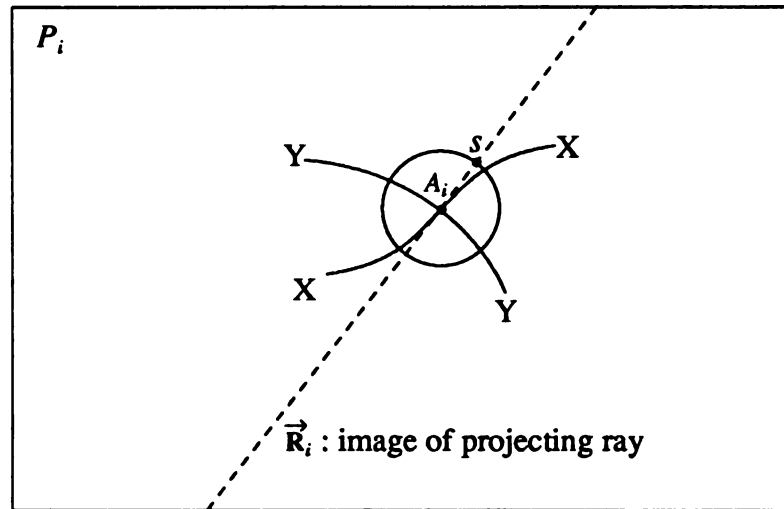


Figure 3.7 X- and Y-stripes distinguished w.r.t. \vec{R}_i

3.4.4 Algorithm 3 — single network solution via constraint propagation

The purpose of this algorithm is to provide possible 3-D surface solutions for a single 2-D stripe network.

input : Network $n=(V, E)$ and a set G_{p_i} of igw-points resulting from algorithm 2 for each image-point $p_i \in V$.

output : Set of surfaces that corresponds to the network n (ambiguous surface solutions).

method : This algorithm is a DFS (depth first search) procedure. During the graph (network) traversal, consistency between neighboring points in the network is evaluated. Consistency criteria are the constraint rules T1-T3. In this algorithm, s, u, v without subscript "i" are image-points, p_s, p_u, p_v denote their corresponding igw-points. A decision must be made as to whether a consistent interpretation for all network points is needed or whether some may be omitted to allow for error or accidents.

- (1) Pick as the starting point an image-point $s \in V$, whose associated set G_s has smallest size. All points in V and edges in E are labeled "undeleted". $O(|V|)$ time is needed.
- (2) Initialize the "output list" $L \leftarrow \emptyset$. Select an unused igw-point p_s from the set G_s of s . If p_s exists, put p_s in L and $u \leftarrow s$; else stop.
- (3) Select an unexamined edge $(u, v) \in E$ where v is not marked "deleted". A igw-point $p_v \in G_v$ is selected so that p_v is compatible with p_u and with other neighbors of p_u according to the constraints T1-T3. If such a p_v is found, $L \leftarrow L \cup \{p_v\}$ and $u \leftarrow v$; else mark v "deleted" from V and mark all edges that involve v "deleted" from E .
- (4) If $u \neq s$ go to (3) else go to (5).
- (5) Reorganize the "output list" L (set of igw-points) into X-stripes and Y-stripes using x_i, y_i of each $p \in L$, and sort these stripes on *stripe#*. Output L as a resulting

surface.

- (6) Unmark all "deleted" flags, go to (2).

In step (3), "deleted" marking is for obtaining partial surface solutions that involves only a subset of V . If solutions involving all points in V are required, no marking is needed; in this case, if propagation is blocked, the algorithm may simply discard the current partial result in L and go back to step (2).

For one surface solution, each image-point in network n is visited once, hence $O(|V|)$ time is needed. The total number of surface solutions for n is no more than $|G_s|$. So the algorithm takes $O(|V| \cdot |G_s|)$ time, where $|V|$ is the number of image-points in a network, $|G_s|$ is the size of the smallest set of igw-points associated with image-points in the network.

3.4.5 Algorithm 4 (Turning-Right algorithm) — network boundary extraction

The purpose of this algorithm is to locate the extremities of each imaged surface element. It does so by a common technique of border following.

input : Network $n=(V, E)$.

output : Ordered set of image-points $B= \{p_{ij}, j=1,..., |B| \}$ such that it forms the boundary of the network in the image.

method : Since all points of interest in this algorithm are in the image plane, we drop the subscript "i".

In a stripe network, points of degree 1 (stripe end points) and degree 4 (stripe intersections) are easy to handle, because degree-1 points are always on network borders, while degree-4 points are always interior to the stripe networks. We only need to worry about points of degree 2 and 3. They may be on the boundary, but it is difficult to get their order correct on the boundary as illustrated in Figure 3.8(a). To overcome this problem, we extend degree-2 and

degree-3 points to degree 4 as in the steps (1) and (2). Then all points are of degree 1 and degree 4 and are handled in an uniform way.

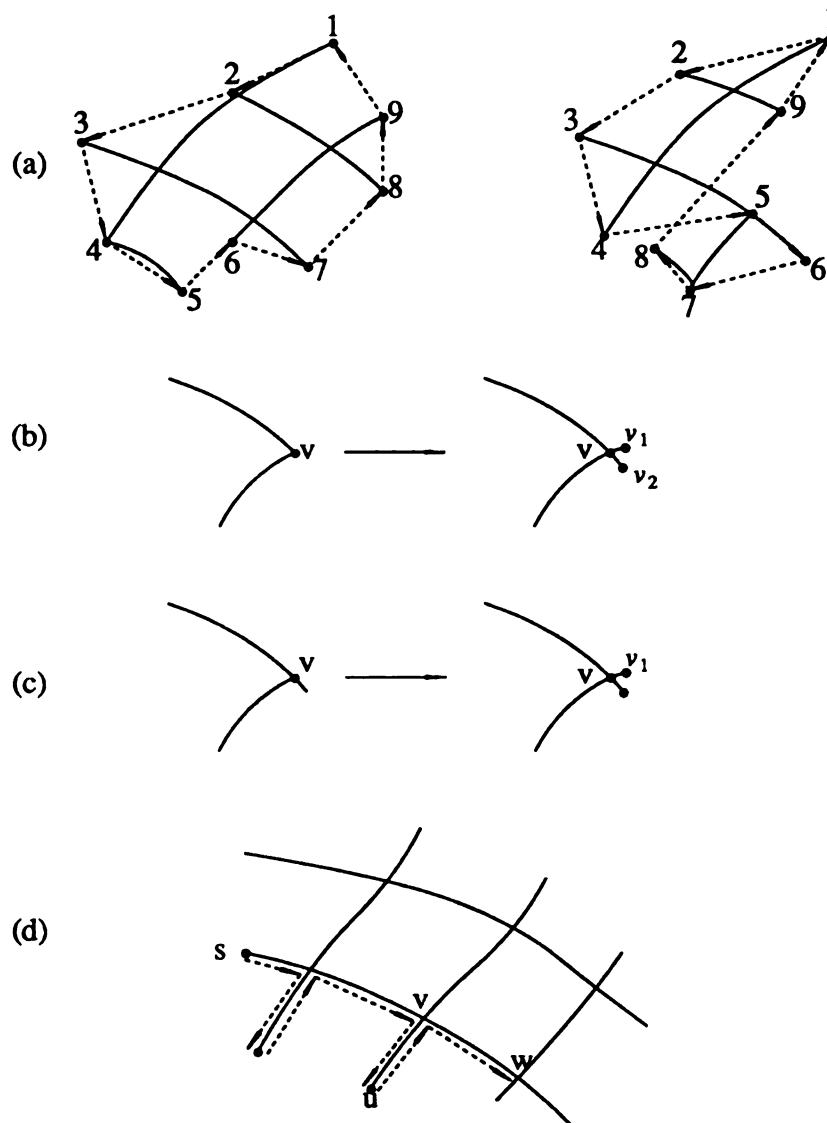


Figure 3.8 **Algorithm 4 (Turning-Right Algorithm)**

- (a) The order of points on boundary (dashed lines) may not be correct at points of degree 2 and 3.
- (b) Extend point of degree 2 to degree 4, v is the "mother" of v_1 and v_2 .
- (c) Extend point of degree 3 to degree 4, v is the "mother" of v_1 .
- (d) Find boundary of a network using the "turning right" algorithm.

- (1) For each image-point $v \in V$ of degree 2 (where an X-stripe and a Y-stripe meet but do not cross, i.e. a corner), we "extend" it to degree 4 by adding two degree-1 points v_1 and v_2 to V , and two edges (v, v_1) and (v, v_2) to E such that v_1 and v_2 are only 1-pixel off v and in the directions extended by the original two edges of v . Call v the "mother" of v_1 and v_2 . See Figure 3.8(b).
- (2) For each $v \in V$ of degree 3, "extend" it to degree 4 by adding a degree-1 point v_1 to V and an edge (v, v_1) to E such that v_1 is only 1-pixel off v and in the direction extended by one of the original three edges of v , determined by Lemma 3.3. Call v the "mother" of v_1 . See Figure 3.8(c). Steps (1) and (2) examine only those points of degree 2 and 3 that are a small fraction of the total number of points in V . The time needed is $O(|V|)$.
- (3) Initialize $B \leftarrow \emptyset$. Arbitrarily pick an image-point $u \in V$ of degree 1 as the starting point. $B \leftarrow B \cup \{u\}$. Let u 's neighbor be v .
- (4) If v is of degree 1, if v has "mother" v' then $B \leftarrow B \cup \{v'\}$, else $B \leftarrow B \cup \{v\}$. If v is of degree 4, let w be the rightmost neighbor of v with respect to the direction $\vec{u}\vec{v}$, and $u \leftarrow v, v \leftarrow w$. See Figure 3.8(d). It takes $O(1)$ time.
- (5) If $v =$ starting point, stop; else go to (4). The loop (4)-(5) takes $O(|V|)$ time, since each point in V is visited at most once.

The time complexity of this algorithm is $O(|V|)$. For large stripe networks, the time cost is much less since many interior grid points are never visited.

3.4.6 Algorithm 5 — solution for the scene

This algorithm eliminates incompatible sets of individual surface solutions by enforcement of the non-overlapping rule G3.

input : Set of networks $N = \{n_j, j=1, \dots, |N|\}$, associated with each n_j a set S_j of surfaces obtained from algorithm 3, the boundary of each surface resulting from

algorithm 4, and the projector calibration matrix M_P .

output : Set of scenes, i.e. set of compatible combinations of surfaces, each surface being associated with one network n_j .

method : This algorithm combines surfaces (one surface for each network) to form a scene. A combination is valid only if all surfaces in the combination have non-overlapping projections in the projector plane, according to the geometric constraint G3 and the topological constraints T4 and T5 stated in section II.

- (1) For each network n_j , project the boundary of each surface in the associated set of n_j onto the projector plane, using the projector calibration matrix M_P . Now we have for each network n_j a set of regions in the projector plane $R_j = \{r_{jk}, k=1, \dots, |R_j|\}$.
- (2) Select one region r_{jk} from each set R_j at a time to form a combination $C = \{r_{jk}, j=1, \dots, |N|, 1 \leq k \leq |R_j|\}$. C is declared invalid and discarded as soon as overlapping between regions in C is found (violation of the constraint G3, under the smoothness and continuity assumptions T4 and T5). If C is valid and completely constructed, output C as one possible scene. Note that the complete set C needs not to be constructed if it is found invalid during its construction. In the worst case the time cost is $T = O(\sum |R_j| |r_{jk}|)$.
- (3) Repeat step (2) until all combinations are exhausted. The worst case time complexity is $O(\prod T |R_j|)$, where T is the previous worst case time cost in step (2). But, since we are able to terminate expansion of combination C of regions at an early stage when C is found invalid, just like tree pruning, the average time needed is less.

3.5 A Complete Example

A series of experiments have been done in the PRIP lab; the results show that the above algorithms do produce good surface solutions with a small degree of ambiguity. Here by "good" we mean accuracy — an average of about 1mm (worst 3-4mm) error in 3-D, with the camera stand-off a little less than 1 meter. This performance is almost as good as humans' eyes. Because only sparse data are processed, the algorithms should run reasonably fast — for a 512×512 image, usually only a couple of hundred or up to a thousand points need to be processed rather than a quarter million pixels. Because we didn't put much effort on the efficiency of the algorithms, they run a little slower than they should have. For a 512×512 image, the algorithms produced surface solutions in about 5-10 minutes on a VICOM image processing system and VAX/8600, including reading-writing files between algorithms.

Let's go through the details, using an example, of how the surface solutions are developed. A striped image of a scene consisting of a block, a Coke can and a diesel piston is shown in Figure 3.9 (reprint of Figure 2.2).

After applying algorithm 1, image points are detected and stripe networks are extracted. Each image point is assigned a unique id. The detected network points of the image are given in Figure 3.10. Notice that the stripes on the piston were not clearly seen due to degraded illumination in the area of the piston, and the stripe network on the piston surface is fragmented into small pieces. This was because we were not very careful with the illumination condition when we ran the experiment. The results reported here include only those for the block and the Coke can.

Algorithm 2 computes a set of candidate grid-points for each image point. Most of the sets have 2 to 7 candidate points, a few sets have more or less candidates. The result for the upper surface of the block is shown in Figure 3.11.

The constraint propagation algorithm (algorithm 3) computed network-consistent surface solutions for each network in the image. Two possible solutions of the upper

surface of the block resulted, which are marked A and B in Figure 3.11. The detailed 3-D results are given in Table 3.1. There are also two network-consistent solutions for the lower surface of the block, two for the Coke can.

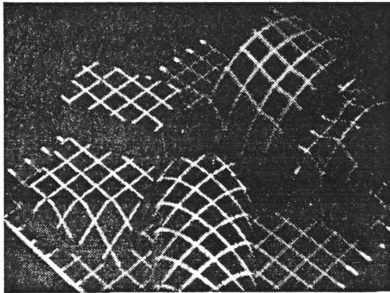


Figure 3.9 A striped image of a block, a Coke can, and a piston

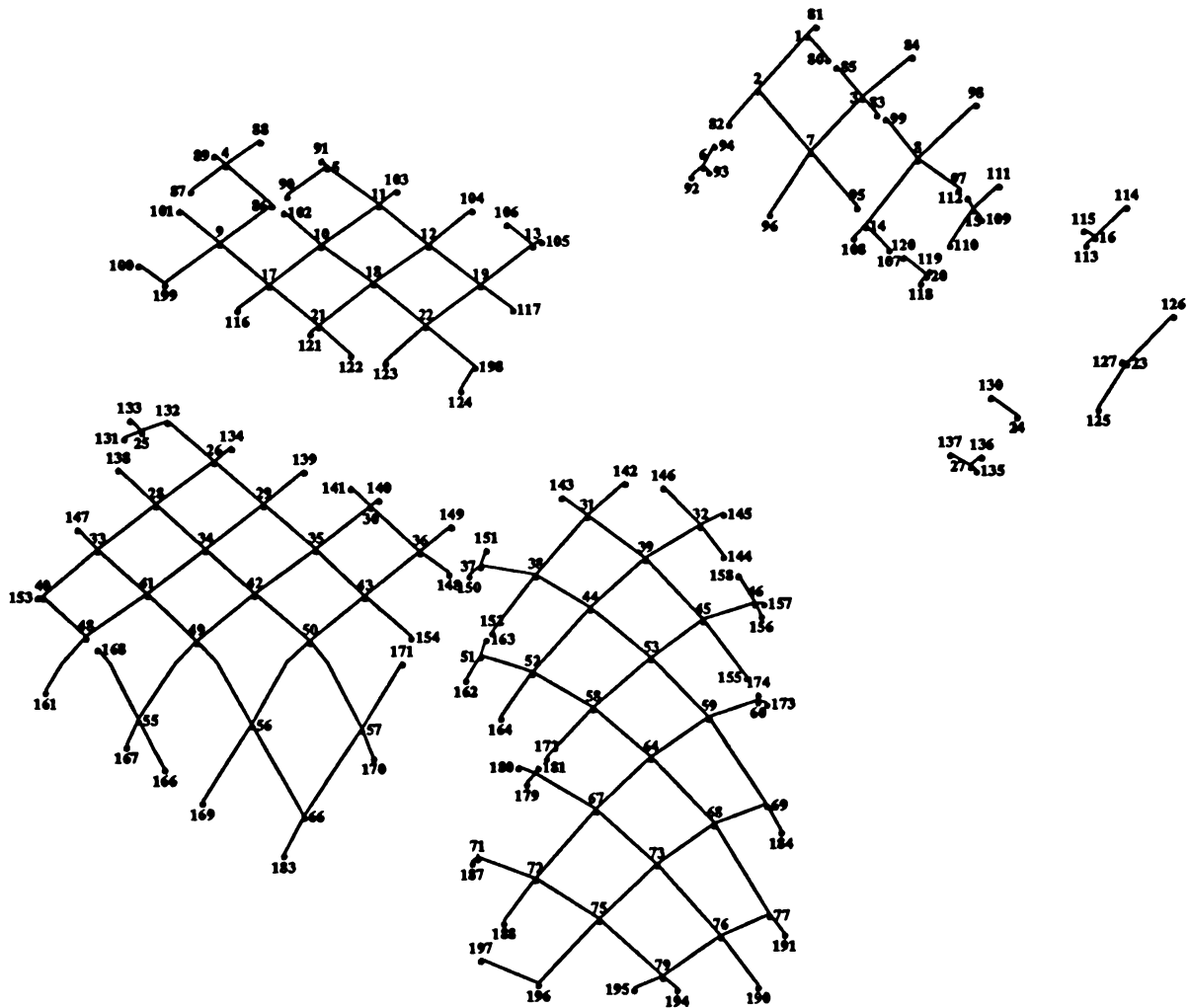
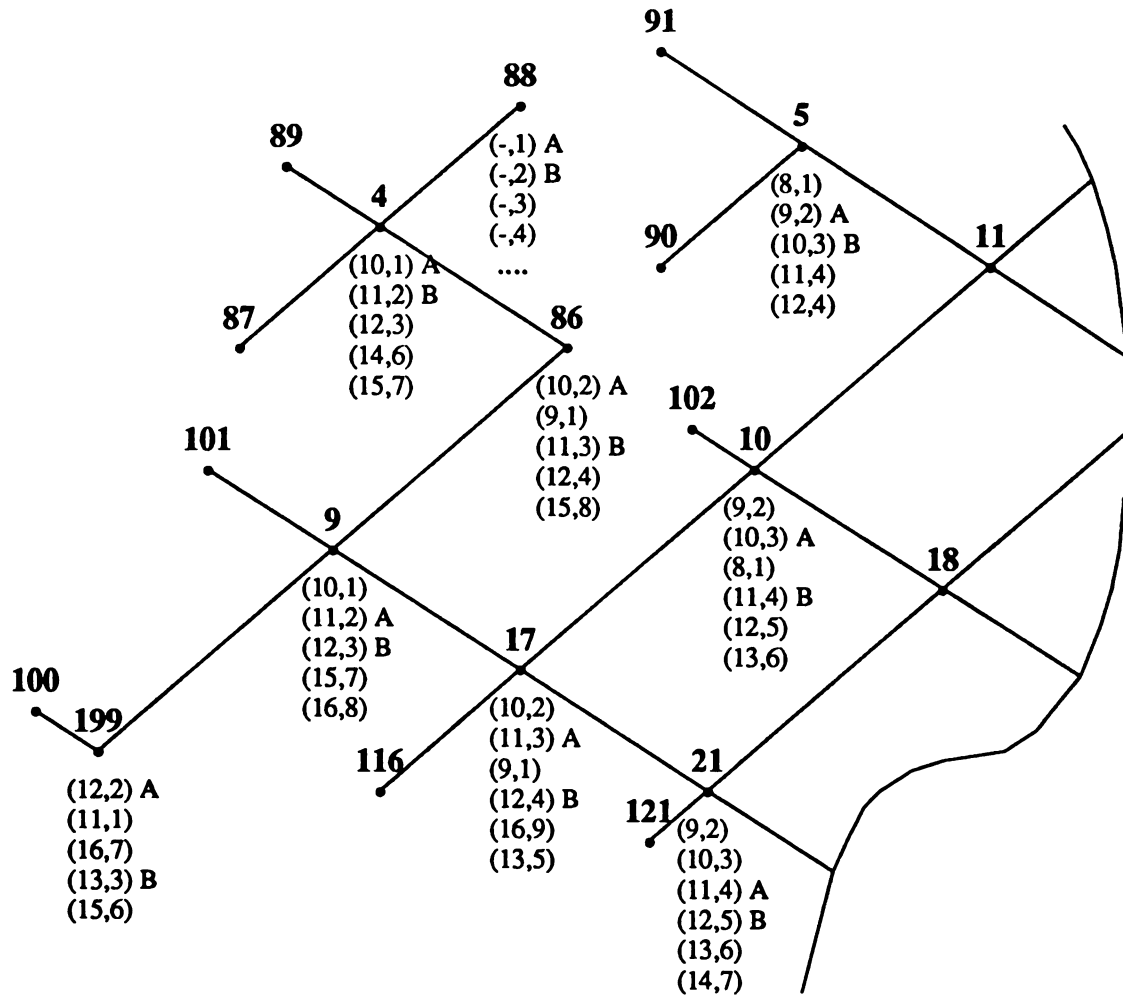


Figure 3.10 detected grid points in the jumble image



A : first solution

B : second solution

Figure 3.11

Candidate labels for some grid points of the upper surface of the block. Results for some stripe end-points are not shown. Only two network-consistent solutions (A and B) for this surface are feasible, as the result of algorithm 3.

Table 3.1 Two solutions for the upper surface of the block

Solution	Network Point No.	Grid		3D coordinate		
		X _g	Y _g	X	Y	Z
A						
	4	10	1	151.5	34.7	80.3
	5	9	2	136.4	49.7	79.6
	9	11	2	166.3	49.7	79.9
	10	10	3	151.6	64.4	79.3
	11	9	3	135.9	65.2	79.7
	12	9	4	136.0	80.0	79.4
	13	8	5	120.9	95.1	79.5
	17	11	3	166.2	64.7	79.1
	18	10	4	151.2	79.7	79.9
	19	9	5	135.9	95.1	79.4
	21	11	4	166.3	79.6	79.4
	22	10	5	151.1	94.8	79.5
	86	10	2	151.6	49.4	79.6
	87	-	1	160.8	34.1	79.2
	88	-	1	141.2	34.1	79.4
	89	10	-	152.0	31.7	81.1
	90	-	2	147.3	49.1	78.9
	91	9	-	137.0	48.8	81.4
	100	12	-	181.9	42.3	79.9
	101	11	-	166.9	37.7	80.9
	102	10	-	151.9	53.5	80.8
	103	-	3	129.8	64.0	78.5
	104	-	4	122.5	79.0	78.7
	105	-	5	117.1	93.8	77.0
	106	8	-	122.0	88.8	81.6
	116	-	3	174.3	64.0	78.1
	117	9	-	137.0	105.3	81.5
	121	-	4	168.1	78.9	78.3
	122	11	-	166.9	89.7	79.8
	123	-	5	161.5	93.8	76.8
	124	-	6	155.1	108.8	75.8
	198	10	6	151.5	109.3	80.0
	199	12	2	181.2	49.7	79.7

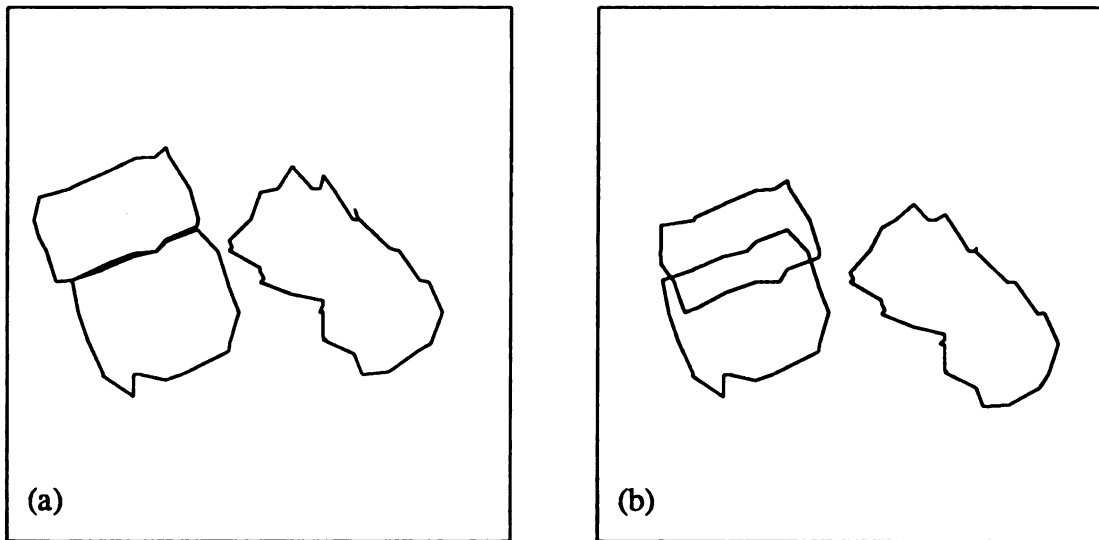
Solution	Network Point No.	Grid		3D coordinate		
		X _g	Y _g	X	Y	Z
B						
	4	11	2	164.5	52.3	102.0
	5	10	3	150.2	66.6	101.5
	9	12	3	179.0	67.1	102.4
	10	11	4	165.1	81.1	101.9
	11	10	4	150.0	81.6	101.9
	12	10	5	150.3	95.9	102.0
	13	9	6	136.0	110.2	102.1
	17	12	4	179.1	81.6	102.1
	18	11	5	165.0	95.9	102.9
	19	10	6	150.4	110.4	102.4
	21	12	5	179.4	96.0	102.7
	22	11	6	165.1	110.4	102.9
	86	11	3	164.9	66.6	101.8
	87	-	2	172.6	50.7	99.7
	88	-	2	154.0	50.7	99.9
	89	11	-	166.0	50.8	104.5
	90	-	3	160.2	65.4	100.1
	91	10	-	151.4	66.4	104.0
	100	13	-	195.2	62.2	105.1
	101	12	-	180.6	57.2	105.1
	102	11	-	166.0	71.4	104.2
	103	-	4	143.9	80.1	100.3
	104	-	5	137.4	94.8	101.2
	105	-	6	132.8	109.4	100.4
	106	9	-	136.7	103.8	103.5
	116	-	4	186.2	80.1	99.9
	117	10	-	151.3	119.9	104.1
	121	-	5	180.7	94.8	100.8
	122	12	-	180.6	106.2	104.1
	123	-	6	174.9	109.4	100.2
	124	-	7	169.3	124.1	100.0
	198	11	7	165.7	124.4	103.7
	199	13	3	193.3	67.3	102.5

Note : From the model and pose, it is known that surface points (X,Y,Z) should satisfy the planar equation $Z = 80\text{mm}$.

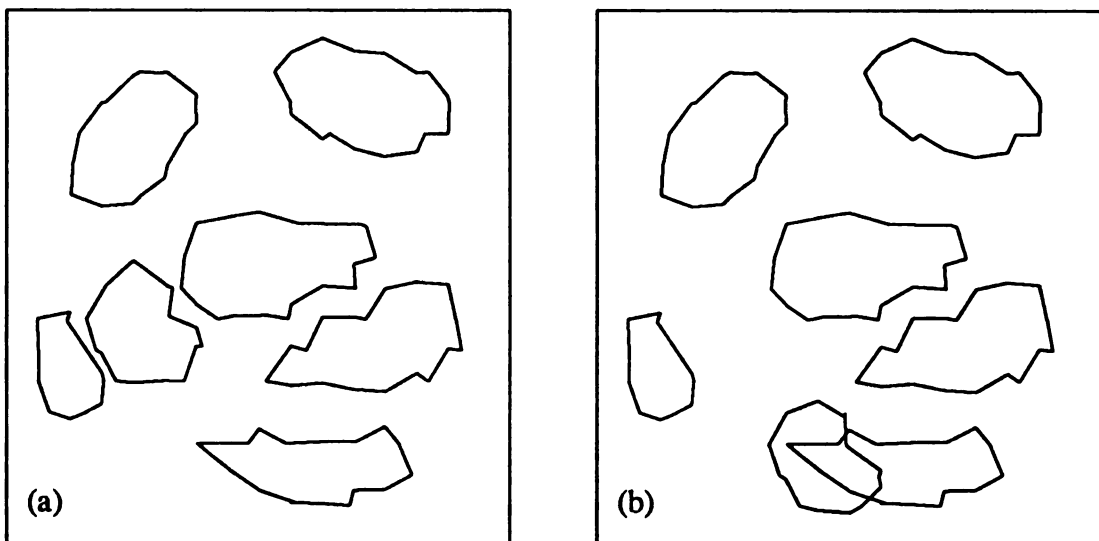
The network boundaries are extracted by the ‘‘Turning Right’’ algorithm (algorithm 4). Algorithm 5 checks the validity of combinations of surface solutions. Since there are two solutions for each of the three networks in the image (upper surface and lower surface of the block, the Coke can), we come up with eight possible combinations. The non-overlapping rule culls six of the eight possibilities, and leaves two as the final results. The upper panel of Figure 3.12 shows a valid and an invalid combination of surface solutions of the jumble.

Several other images were also processed : (1) a clay sculpture of a cobra, (2) a diesel piston, and (3) seven potatoes. We have seen these images in Figure 2.2. The backprojection of two combined solutions for the potato image is shown at the lower panel of Figure 3.12. Table 3.2 illustrates the results for these images.

From the experiments we see that (1) large image networks have fewer surface solutions, sometimes a unique solution (as in the case of the cobra sculpture), because of tightened constraints caused by a large number of image points; (2) the uniqueness constraint is very powerful on images of multiple networks, as in the case of the potatoes where only 5 combinations of surface solutions survived among a total of 36 possibilities — an 86% cut; (3) ambiguity still remains in the final results, but to a much smaller degree, and it can be sorted out by a little additional knowledge such as more assumptions about the real world (for example, in the 5 solutions of the potatoes image, four of them have potatoes floating in the air), by information from a different viewing position, or by use of object models.



backprojection of some combined solutions for the jungle image



backprojection of some combined solutions for the potato image

Figure 3.12 (a) valid combination (b) invalid combination

Table 3.2 Summary of surface solutions for the jumble, cobra, piston and potato images

image	surface patches	number of possible surface solutions	number of surface combinations after applying uniqueness constraint
jumble	upper	2 (A,B)	2 (AAA,BBA)*
	lower	2 (A,B)	
	Coke can	2 (A,B)	
cobra	head	3 (A,B,C)	1 (BA)
	body	1 (A)	
potatoes	P_1	1 (A)	5 solutions : (ABAACAA, ABACCAA, ACABCAA, ACAACAA ACACCAA)
	P_2	4 (A,B,C,D)	
	P_3	1 (A)	
	P_4	3 (A,B,C)	
	P_5	3 (A,B,C)	
	P_6	1 (A)	
	P_7	1 (A)	

* Solution BBA represents the combinations of solution B of the upper surface, solution B of the lower surface, and the solution A of the Coke can.

3.6 Quantitative Analysis on the Degree of Ambiguity

The degree of ambiguity refers to the number of surface solutions in 3-space that survive the constraint test. We want the number of surface solutions to be small. If the number is one, we get a unique solution. The question is, without any knowledge about the objects in the scene except the projection and imaging geometry, how many solutions may be possible? Figure 3.13 illustrates the situation. We see that the number of possible surface solutions depends on several factors, for instance, light stripe spacing, number of stripes, and the distance from the camera to the scene. In addition, a camera ray and a projector ray (both are lines in 3D) may not intersect in space even though the projector ray does create that very point in the 2D image, due to measurement and computation inaccuracy. We treat a camera ray and a projector ray as if they did intersect in space if the distance between the two rays is less than a predetermined tolerance T . Taking this factor into consideration, the degree of ambiguity is also an increasing function of T .

Can the degree of ambiguity be arbitrarily large, if infinitely many projected light stripes are assumed? We shall show quantitatively that this will usually not be the case. The analysis is based on two projection models : orthographic and perspective.

3.6.1 Orthographic projection

3.6.1.1 Assumptions

Under an orthographic, or parallel projection, model the projector projection is parallel. That is, all X-stripes are parallel to each other, all Y-stripes are parallel to themselves. Camera imaging is assumed perspective. We are to examine how many locations are possible in 3-space for a square-shaped surface patch on the xy plane (worktable) formed by stripes whose stripe numbers are X_g, X_g+1, Y_g , and Y_g+1 . The projection model is illustrated in Figure 3.14.

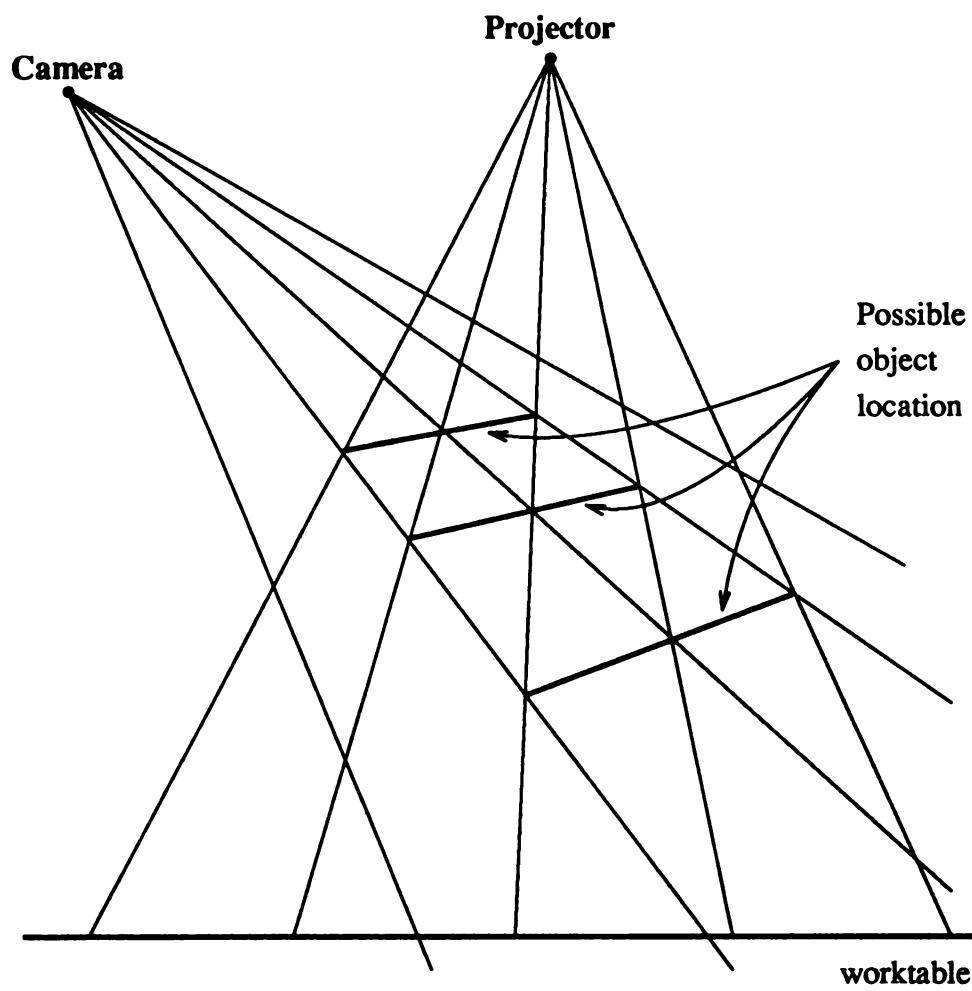


Figure 3.13 Ambiguity of location in 3-space

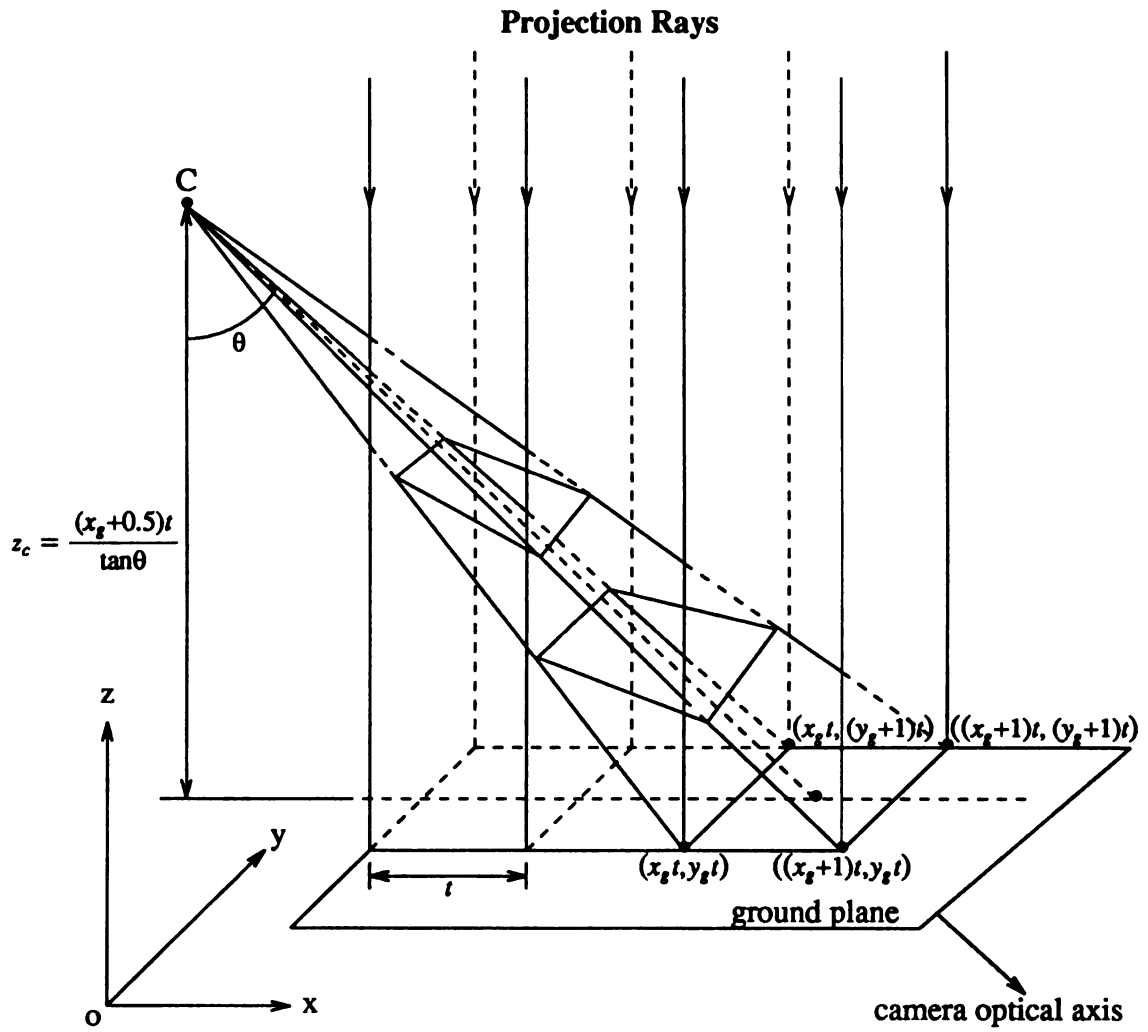


Figure 3.14 Parallel projection model

3.6.1.2 Basic formulas

The equation of the line through two fixed points (x_1, y_1, z_1) and (x_2, y_2, z_2) is given by

$$\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} = \frac{z-z_1}{z_2-z_1} \quad (1)$$

The *direction numbers* of a line

$$Ax + By + Cz + D = 0$$

$$ax + by + cz + d = 0$$

are l, m, n given by

$$l = \begin{vmatrix} B & C \\ b & c \end{vmatrix}, \quad m = \begin{vmatrix} C & A \\ c & a \end{vmatrix}, \quad n = \begin{vmatrix} A & B \\ a & b \end{vmatrix}$$

The distance between two lines is

$$d = \left| \frac{(x_1-x_2)L + (y_1-y_2)M + (z_1-z_2)N}{\sqrt{L^2 + M^2 + N^2}} \right| \quad (2)$$

where (x_1, y_1, z_1) is an arbitrary point one line and (x_2, y_2, z_2) is an arbitrary point on another line, and

$$L = \begin{vmatrix} m_1 & n_1 \\ m_2 & n_2 \end{vmatrix}, \quad M = \begin{vmatrix} n_1 & l_1 \\ n_2 & l_2 \end{vmatrix}, \quad N = \begin{vmatrix} l_1 & m_1 \\ l_2 & m_2 \end{vmatrix}$$

with m_1, n_1, l_1 the direction numbers of the first line, and m_2, n_2, l_2 the direction numbers of the second line.

3.6.1.3 Computation in 3-space

Let stripe spacing be t , the four vertices of the surface square be $(x_s t, y_s t, 0)$, $((x_s+1)t, y_s t, 0)$, $((x_s+1)t, (y_s+1)t, 0)$, $(x_s t, (y_s+1)t, 0)$. Let the optical axis of the camera be from the camera $(0, y_c, z_c)$ to the center of the surface square, i.e. the coordinates of the camera are

$$\left[0, (y_s + \frac{1}{2})t, \frac{(x_s + \frac{1}{2})t}{\tan \theta} \right]$$

The camera ray has the equation

$$L_1 : \begin{cases} x + \frac{2x_g y}{(2x_g+1)y - \tan\theta z} - (2y_g+1)x_g t = 0 \\ (2x_g+1)y - \tan\theta z - (2x_g+1)y_g t = 0 \end{cases}$$

Thus, the direction numbers for L_1 are obtained as

$$l_1 = \begin{vmatrix} 2x_g & 0 \\ 2x_g+1 & -\tan\theta \end{vmatrix} = -2x_g \tan\theta, \quad m_1 = \begin{vmatrix} 0 & 1 \\ -\tan\theta & 0 \end{vmatrix} = \tan\theta, \quad n_1 = \begin{vmatrix} 1 & 2x_g \\ 0 & 2x_g+1 \end{vmatrix} = 2x_g+1$$

Suppose k solutions exist, i.e. the k -th solution is valid, which is obtained by shifting the square surface patch towards the camera along its optical axis by k X-stripes, then the corresponding equation of the projector ray is given by

$$L_2 : \begin{cases} x - (x_g - k)t = 0 \\ y - y_g t = 0 \end{cases}$$

and the direction numbers are

$$l_2 = \begin{vmatrix} 0 & 0 \\ 1 & 0 \end{vmatrix} = 0, \quad m_2 = \begin{vmatrix} 0 & 1 \\ 0 & 0 \end{vmatrix} = 0, \quad n_2 = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1$$

Hence, the distance between the camera ray L_1 and the projector ray L_2 is computed according to formula (2) :

$$d = \frac{((x_g t) - (x_g - k)t) \begin{vmatrix} \tan\theta & 2x_g+1 \\ 0 & 1 \end{vmatrix} + 0 + 0}{\sqrt{\tan^2\theta + 4x_g^2 \tan^2\theta}} = \frac{kt}{\sqrt{1+4x_g^2}}$$

Making this distance less than a preassigned tolerance T , we have

$$k \leq \frac{\sqrt{1+4x_g^2}}{t} T \approx \frac{2x_g}{t} T$$

Since L_1 is the camera ray (among the 4 camera rays) that has the greatest distance from the projector ray, we do not need to consider the other 3 rays. Hence, we conclude that the number k of possible locations of a surface patch in 3-space is proportional to the camera standoff (determined by x_g) and the tolerance T , inversely proportional to the stripe spacing t .

3.6.1.4 Computation in xy-plane

To make things simpler, we just look at the projection of these rays in the xy-plane. Because of the parallel projector projection, the distances between a camera ray and projector ray is the same as its xy-plane projection.

Suppose that the camera is at $(0, y_c)$, and the four vertices of the square-shaped surface patch are $(x_g t, 0), ((x_g + 1)t, 0), ((x_g + 1)t, t)$, and $(x_g t, t)$. The 4 camera rays are

$$l_1 : y_c x + x_g t y - x_g t y_c = 0$$

$$l_2 : y_c x + (x_g + 1)t y - (x_g + 1)t y_c = 0$$

$$l_3 : (y_c - t)x + (x_g + 1)t y - (x_g + 1)t y_c = 0$$

$$l_4 : (y_c - t)x + x_g t y - x_g t y_c = 0$$

The distance from point (u, v) to line $Ax + By + C = 0$ is

$$d = \left| \frac{Au + Bv + C}{\sqrt{A^2 + B^2}} \right|$$

The i -th possible location of the surface patch in 3-space (moving the patch towards the camera i x-stripes) would have 4 vertices $((x_g - i)t, jt), ((x_g - i + 1)t, jt), ((x_g - i)t, (j + 1)t)$, and $((x_g - i + 1)t, (j + 1)t)$, where j is the number of y-stripes the surface patch should move to the camera in order to obtain the i -th location.

These 4 points will have distances from $l_k, k=1, 2, 3, 4$ as follows

$$d_1 = \frac{t |x_g t j - y_c i|}{\sqrt{y_c^2 + (x_g t)^2}}$$

$$d_2 = \frac{t |(x_g + 1)t j - y_c i|}{\sqrt{y_c^2 + ((x_g + 1)t)^2}}$$

$$d_3 = \frac{t |(x_g + 1)t j - y_c i + it|}{\sqrt{(y_c - t)^2 + ((x_g + 1)t)^2}}$$

$$d_4 = \frac{t |x_g t j - y_c i + it|}{\sqrt{(y_c - t)^2 + (x_g t)^2}}$$

In order to make the i -th solution valid, $d_k(k=1,2,3,4)$ must \leq tolerance T at the same time.

Let us consider some special cases where the camera viewing direction varies.

(1). $\theta=\pi/4$

In this case, $y_c=x_g t$, $i=j$. d_3 is the largest :

$$d_3 = \frac{\sqrt{2}+i}{x_g} \Rightarrow i \leq \frac{T x_g}{\sqrt{2} t}$$

(2). $\theta=0$

Here the optical axis is parallel to y -stripes, i.e. $y_c=0$, $j=0$. Computation shows that

$d_4=\max(d_k), k=1,2,3,4$:

$$d_4 = \frac{it}{x_g} \Rightarrow i \leq \frac{T x_g}{t}$$

(3). $\tan\theta=2$, i.e. $\theta=27^\circ$, or $y_c=0.5x_g t$, and $i=2j$.

In this case, all candidate solutions by shifting the surface patch towards the camera odd number of x -stripes are considered invalid, since the distance calculated is about half of the stripe spacing, which is usually considered too large. By shifting the patch to the camera even number of x -stripes, say $2n$, we get the largest distance as

$$d_3 = \frac{6jt}{\sqrt{5x_g^2+4x_g+8}}$$

Make $d_3 \leq T$, we get

$$j \leq \frac{\sqrt{5x_g^2+4x_g+8} T}{6t}$$

where $i=2n=2j$.

According the above analysis, we obtain Table 3.3 for the number of possible locations in 3-space of a square-shaped surface patch on the ground plane, with a tolerance $T=3.5\text{mm}$.

Table 3.3 Number of possible locations in 3-D for a striped square, assuming parallel projection.

camera angle	camera stand-off (mm)	stripe spacing (mm)					
		5	10	15	20	25	30
0°	300	14	7	5	3	2	2
	450	21	10	7	5	4	3
	600	28	14	9	7	5	4
27°	300	5	2	1	1	1	1
	450	8	4	2	2	1	1
	600	10	5	3	2	2	1
45°	300	10	5	3	2	2	1
	450	15	7	5	3	2	1
	600	20	10	6	5	4	3

In our experiments, the stripe spacing is 15mm, camera is at about 450–600mm from the scene with viewing angle about 45°, the number of possible solutions for a single square is 5–6. For a larger network (more constrained), which is usually the case, the number of possible solutions will be less.

3.6.2 Perspective Projection

3.6.2.1 Assumption

Under the perspective model, both camera imaging and projector projection are perspective. We will expect larger ambiguity than in parallel model, since both camera rays and projector rays are tapering and getting closer and closer to each other. Thus there is a

good chance for a camera ray to “intersect” a projector ray in 3-space — the distance between the two rays will be easily smaller than a predetermined tolerance T .

3.6.2.2 Computation

We use the similar set-up as in the previous section. The coordinate system is the same. The camera position is at $(0, y_c, z_c)$. and the projector is at (x_p, t, z_p) . That is, the optical axis of the projector is vertically from the center of the projector to the point (x_p, t) , which is assumed to be one of the vertices of the square patch on the worktable. The surface patch has 4 vertices $(x_p, t, 0), ((x_p+1)t, 0), ((x_p+1)t, t)$ and (x_p, t) , as shown in Figure 3.14.

Using the same formulas as in the orthographic projection case, it turns out that the analytical expressions are too messy to be useful. Instead of obtaining explicit analytical form for ray-distance, we just run a computer program to compute it numerically on different set of parameters. The algorithm would be easier to read in vector notations.

Let \mathbf{a}, \mathbf{b} be two points on line l_1 , and \mathbf{c}, \mathbf{d} be two points on line l_2 . Then, the $\mathbf{u}=\mathbf{a}-\mathbf{b}$ and $\mathbf{v}=\mathbf{c}-\mathbf{d}$ are vectors on l_1 and l_2 . The unit common normal $\mathbf{n}=(\mathbf{u} \times \mathbf{v})/|\mathbf{u} \times \mathbf{v}|$ is perpendicular to both \mathbf{u} and \mathbf{v} . Take a vector whose head is on l_1 and tail on l_2 , say $\mathbf{w}=\mathbf{c}-\mathbf{a}$, and the projection of this vector onto the common normal will be the distance between l_1 and l_2 . That is, $d=\langle \mathbf{w}, \mathbf{n} \rangle$. This algorithm yields the Table 3.4.

From Table 3.4 we can see that if the camera is viewing at 45° there might be many surface solutions possible, since the camera rays will intersect the projector rays at every shift. If the camera is far away from the surface patch, the number of possible surface locations is also large, because the tapering camera rays and projector rays will be getting very close to each other so the ray-distance will be $\leq T$. Also, it is obvious that the number of possible surface locations decreases as the stripe spacing increases.

The above analysis is based on a single quadrilateral on the $Z=0$ plane. It is conceivable that for a scene of multiple objects each is much larger than a single quadrilateral,

Table 3.4 Number of possible locations in 3-D for a striped square, assuming perspective projection.

camera angle	distance from camera to scene		stripe spacing (mm)		
	horizontal	vertical	10	15	20
32.5°	550	700	37	11	4
45°	550	700	42	19	8
32°	400	700	23	9	4
45°	400	700	30	13	3
31°	250	700	13	5	1
45°	250	700	18	8	3
23°	350	700	18	7	1
45°	350	700	26	7	5
63.4°	100	650	11	5	1
45°	200	650	13	4	1
26.6°	100	650	4	2	1
45°	100	650	5	3	0

the resulting surface solutions are expected to be much less ambiguous. This is confirmed in our experiments.

3.7 Conclusions

This chapter dealt with 3-dimensional sensing problems using a projected grid. The geometric computation for the surface points follows a stereo model — the camera-projector pair. The difficult correspondence problem in the stereo model is simplified to the line identification problem in the structured lighting environment, which can be partially solved through the propagation of certain general constraints. In controlled environments, such as for industrial robots, this approach may provide an economical solution to 3-D sensing.

The discussion and experimental results in this chapter can be outlined as follows.

(1) The 3-D location of a point is computed by intersecting the projecting ray and the imaging ray in 3-space — a typical triangulation method. In practice, the exact intersection may not be found, instead, it is approximated by the least squares solution overdetermined by the four equations (3.10) ~ (3.13).

(2) Because the projecting ray is non-deterministic, there may be a set of possible solutions, each associated with one possible projecting ray (or grid point). All candidate grid points are within a narrow band along the epipolar line of the imaging ray.

(3) The uniqueness and the continuity constraints are from the real world. The deduced geometric and topological rules are very powerful in developing surface solution and reducing ambiguity. Although the result of constraint propagation is still not unique in general, it usually has only a small degree of ambiguity that may further be reduced using knowledge from other sources.

(4) The ambiguity in surface solution represents several possible locations of a surface patch in 3-space, each one of them is equally useful in inferring surface shape and other geometric properties.

CHAPTER 4

Inference of Surface Shape

3-D measurement by itself is not the goal of a vision system. The goal is to understand the scene — to tell what and where the objects are in the scene, i.e. their locations, orientations, shapes, as well as their spatial relationships. The collected 3-D data are useful only for this purpose. The 3-D data obtained from light striping are sparse but not individually isolated, they are spatially related through the stripe networks. How surface shape is inferred from 3-D grid point data and the stripe networks is the focus of this chapter. The objective is to get symbolic labels on surface data. In Section 4.1, a surface shape classification method will be discussed, which is based on the intrinsic Gaussian curvature at various points on the surface. In order to compute the Gaussian curvature, or more specifically, to compute the partial derivatives of the surface function, a discrete B-spline fitting is applied to the 3-D grid data. Experiments show that this approach can work on our striped data, as it has worked on other type of data, range data for example [Besl and Jain 85]. Section 4.2 is concerned with the problems in inferring surface shapes from the 2-D stripe networks in the image. The 2-D stripe networks are considered as stripe textures on which a structural texture approach is applied.

4.1 Surface fitting and local curvatures

Given a set of points in n -dimensional space, it is often desirable to estimate the position of other points in the space in general, or to fit curves ($n=2$) or surfaces ($n=3$) in particular. In our case, we have had certain points in 3-space computed (grid points), and we want to infer the shape of surface on which these 3-D points lie. An analytical form of a surface is usually hard to get except for those that are planar, spherical, cylindrical and the like. A common way to describe a surface is to do so *locally*, in terms of surface normal or curvatures. Curvature properties, especially the intrinsic Gaussian curvature, has been studied for many years, and used in surface segmentation and classification for both intensity and range images [Ittner and A. K. Jain 85, Besl and R. Jain 85, Medioni and Nevatia 84, Laffey, Haralick and Watson 82, Ponce and Brady 85, Yang and Kak 86, Besl and R. Jain 86]. In the case of striped data, B-spline surface fitting followed by Gaussian curvature computation is feasible because the 3-D grid points available are the control points needed in fitting. First we briefly review some local surface theory in differential geometry [Millman and Parker 77, Lipschutz 69]. Then we describe the approach that computes the Gaussian curvature and the two principal curvatures at surface points based on B-spline surfaces [Yang and Kak 86]. Finally, surface shape is determined based on the clusters of the calculated curvatures.

4.1.1 Differential geometry — local surface theory

Surface tangent plane and normal

A surface $S \in \mathbf{R}^3$ can be expressed in parametric form as

$$\mathbf{x}(u_1, u_2) = (x(u_1, u_2), y(u_1, u_2), z(u_1, u_2))$$

where u_1, u_2 are two parameters and $\mathbf{x} \in S$ can be viewed as a vector from the origin to a point on S .

The partial derivatives of \mathbf{x} with respect to u_1 and u_2 , denoted \mathbf{x}_1 and \mathbf{x}_2 , are

$$\mathbf{x}_i = \frac{\partial \mathbf{x}}{\partial u_i}$$

and the second order partial derivatives are

$$\mathbf{x}_{ij} = \frac{\partial^2 \mathbf{x}}{\partial u_i \partial u_j}$$

$\{\mathbf{x}_1, \mathbf{x}_2\}$ is a basis of the tangent plane $T_{\mathbf{x}}S$ at point \mathbf{x} , i.e. \mathbf{x}_1 and \mathbf{x}_2 span the tangent plane.

The surface normal at point \mathbf{x} is then

$$\mathbf{n} = \frac{\mathbf{x}_1 \times \mathbf{x}_2}{|\mathbf{x}_1 \times \mathbf{x}_2|}.$$

First fundamental form

Since at point p of a surface S the two partial derivatives \mathbf{x}_1 and \mathbf{x}_2 span the tangent plane T_pS , any vector \mathbf{X} in the tangent plane is a linear combination of \mathbf{x}_1 and \mathbf{x}_2

$$\mathbf{X} = a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2 = \sum a_i \mathbf{x}_i$$

The *first fundamental form* on surface S is defined as a bilinear form on T_pS for each $p \in S$, given by

$$\mathbf{I}(\mathbf{X}, \mathbf{Y}) = \sum a_i b_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum a_i b_j g_{ij}$$

where \mathbf{X}, \mathbf{Y} are in the tangent plane T_pS such that

$$\mathbf{X} = a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2 = \sum a_i \mathbf{x}_i, \quad \mathbf{Y} = b_1 \mathbf{x}_1 + b_2 \mathbf{x}_2 = \sum b_i \mathbf{x}_i$$

and $g_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, the inner product of \mathbf{x}_i and \mathbf{x}_j , are called the *metric coefficients* or the *coefficients of Riemannian metric*. The first fundamental form is explicitly written as

$$\mathbf{I}(\mathbf{X}, \mathbf{Y}) = (a_1, a_2) \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

The matrix (g_{ij}) assigns for any $\mathbf{X}, \mathbf{Y} \in T_pS$ their inner product with respect to the basis

$\{\mathbf{x}_1, \mathbf{x}_2\}$. Notice that $g_{12} = g_{21}$.

Second fundamental form

The *second fundamental form* on surface S is a bilinear form on $T_p S$ for each $p \in S$, given by

$$\mathbf{II}(\mathbf{X}, \mathbf{Y}) = \sum a_i b_j \langle \mathbf{x}_{ij}, \mathbf{n} \rangle = \sum a_i b_j L_{ij}$$

where $\mathbf{X}, \mathbf{Y} \in T_p S$ are as before, and $L_{ij} = \langle \mathbf{x}_{ij}, \mathbf{n} \rangle$, the inner product of \mathbf{x}_{ij} and \mathbf{n} , are called the *coefficients of the second fundamental form*. We can also write the second fundamental form as

$$\mathbf{II}(\mathbf{X}, \mathbf{Y}) = (a_1, a_2) \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Notice that $L_{12} = L_{21}$. The geometric meaning of the second fundamental form is that if γ is a unit speed curve on S with tangent \mathbf{T} , the $\mathbf{II}(\mathbf{T}, \mathbf{T})$ defines the *normal curvature* of γ , which measures how the surface normal is changing along the curve γ .

Weingarten map

The Weingarten map \mathbf{W} is, for each $p \in S$, the function $T_p S \rightarrow T_p S$ given by

$$\mathbf{W}(\mathbf{X}) = -\mathbf{X}\mathbf{n}$$

where $\mathbf{X} \in T_p S$, and $\mathbf{X}\mathbf{n}$ is the directional derivative of surface normal \mathbf{n} in the direction of \mathbf{X} . The matrix representation of \mathbf{W} with respect to the basis $\{\mathbf{x}_i, \mathbf{x}_2\}$ is $\mathbf{W} = (w_{lk})$ where

$$w_{kl} = \sum L_{ik} g^{il}$$

and $g^{il} = (i, l)$ -entry of the inverse matrix of (g_{ij}) , which is very easy to compute (for a 2×2 matrix!). More specifically, the matrix of the Weingarten map can be expressed as

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} g^{11} & g^{12} \\ g^{21} & g^{22} \end{bmatrix}$$

Curvatures

To measure how a surface S is curving at a point p , one way is through the normal curvature of curves on S that pass through p . This is aesthetically unsatisfactory because it forces us to break up a surface into infinitely many curves. Rather than to deal with the infinitely many curves, an alternative is to find the maximum and the minimum values of the normal curvature, which are the representatives of the infinitely many values so that they reveal the geometric essence of S at p . A theorem in differential geometry says [Millman and Parker 77]

"At each point of a surface there are two orthogonal directions such that the normal curvature takes its maximum value in one direction and its minimum along the other."

The two orthogonal directions are called *principal directions* at p , the maximum and the minimum values of normal curvatures are called *principal curvatures* of S at p . It turns out that the two principal curvatures are the two eigenvalues of the Weingarten map \mathbf{W} and the two principal directions are the directions of the two corresponding eigenvectors.

Since \mathbf{W} is a linear transformation, there are two associated numerical invariants : the determinant of \mathbf{W} which is the product of the two eigenvalues κ_1 and κ_2 , called the **Gaussian curvature** K ; and one-half of the trace of \mathbf{W} which is one-half of the sum of the two eigenvalues, called the **mean curvature** H

$$K = \kappa_1 \kappa_2 = \det(\mathbf{W})$$

$$H = \frac{1}{2}(\kappa_1 + \kappa_2) = \frac{1}{2}\text{trace}(\mathbf{W})$$

Surface classification based on Gaussian curvature

The surface at point p is categorized into the following types according the Gaussian curvature K at p :

$K > 0$: elliptic (convex or concave)

$K < 0$: hyperbolic (saddle point)

$K = 0$: parabolic if $\kappa_1 \neq 0$ or $\kappa_2 \neq 0$

$K = 0$: planar if $\kappa_1 = \kappa_2 = 0$

4.1.2 Computation — surface fitting using B-splines

The local surface theory outlined above suggests the following simple computational procedure for obtaining curvatures and surface type

1. Find the partial derivatives $\mathbf{x}_1, \mathbf{x}_2$, and the second order derivatives \mathbf{x}_{ij} .
2. Compute normal $\mathbf{n} = \frac{\mathbf{x}_1 \times \mathbf{x}_2}{|\mathbf{x}_1 \times \mathbf{x}_2|}$.
3. Compute the coefficients of the first fundamental form $g_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.
4. Find the inverse matrix $(g^{ij}) = (g_{ij})^{-1}$.
5. Compute the coefficients of the second fundamental form $L_{ij} = \langle \mathbf{x}_{ij}, \mathbf{n} \rangle$.
6. Compute the Weingarten map $\mathbf{W} = (L_{ij}) (g^{ij})$.
7. Compute the eigenvalues κ_1, κ_2 and the eigenvectors of \mathbf{W} .
8. Compute the Gaussian curvature K and the mean curvature H .
9. Classify surface type at point p according to K .

The mean curvature H is not used in classifying surface type, but it is useful for quantitatively describing the extent to which the surface is curving. In fact, H is exactly the average of the normal curvatures of the infinitely many curves at point p over all directions

$$H = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\theta) d\theta$$

where $\kappa_n(\theta)$ is normal curvature along direction θ .

The eigenvectors are not used in surface type classification either, but they provide the orientation information that may be useful in further analysis. For example, if a

surface is determined to be cylindrical because most of the points on the surface are classified so, the orientation of the cylinder (major axis) would be the average direction of the eigenvectors in which the eigenvalues take the value 0 at the various points (the surface does not curve along this direction).

In the computation procedure, step 2 to 9 are very straightforward. The problem is in step 1 — How are these partial derivatives obtained from a set of discrete 3-D points on the surface, which are the only information we have so far? A common approach is to fit an analytic surface locally to a set of points and then take derivatives.

B-spline surface patch

Given 16 points p_{ij} , $i, j = 1, 2, 3, 4$, on a 4×4 equally spaced grid on surface S , such as the ones in Figure 4.1, a B-spline surface patch that fits the middle 4 points is given by

$$\mathbf{x}(u, v) = \mathbf{u} \mathbf{B} \mathbf{P} \mathbf{v}' \quad (4.1)$$

where

$$\mathbf{u} = (u^3 \quad u^2 \quad u \quad 1)$$

$$\mathbf{v} = (v^3 \quad v^2 \quad v \quad 1)$$

$$\mathbf{P} = (p_{ij}), \quad i, j = 1, 2, 3, 4$$

$$\mathbf{B} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

with $0 \leq u, v \leq 1$ being the parameters [Foley and Van Dam 82, Yang and Kak 86]. Point p_{22} is at $u=v=0$; p_{23} is at $u=0$ and $v=1$; p_{32} is at $u=1$ and $v=0$; and point p_{33} is at $u=v=1$. The B-spline patch defines every point in the middle square. For example, the parametric center point would be at $u=v=0.5$.

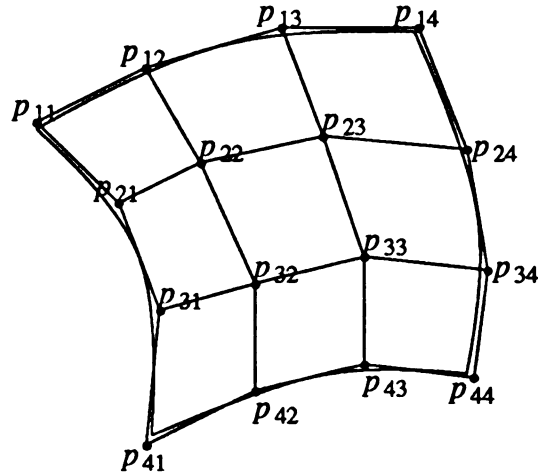


Figure 4.1 B-spline surface patch using 16 control points

Partial derivatives

From the B-spline surface patch (4.1), the partial derivatives at any point in the patch are easy to compute as follows.

$$\mathbf{x}_u = \mathbf{u}' \mathbf{B} \mathbf{P} \mathbf{B}' \mathbf{v}^t$$

$$\mathbf{x}_v = \mathbf{u} \mathbf{B} \mathbf{P} \mathbf{B}' \mathbf{v}^{t'}$$

$$\mathbf{x}_{uu} = \mathbf{u}'' \mathbf{B} \mathbf{P} \mathbf{B}' \mathbf{v}^t$$

$$\mathbf{x}_{uv} = \mathbf{u}' \mathbf{B} \mathbf{P} \mathbf{B}' \mathbf{v}^{t'}$$

$$\mathbf{x}_{vv} = \mathbf{u} \mathbf{B} \mathbf{P} \mathbf{B}' \mathbf{v}^{t''}$$

Since in our experiments the 3-D grid points are in terms of the world coordinate system, a surface patch is considered as a Monge patch $z = z(x, y)$. Hence we take x and y as parameters in substitution of u and v , and the surface patch can be written as $\mathbf{x}(u, v) = (u, v, z(u, v))$. The above partial derivatives are then applied to $z(u, v)$, we obtain

$$\mathbf{x}_u = (1, 0, z_u)$$

$$\mathbf{x}_v = (0, 1, z_v)$$

$$\mathbf{x}_{uu} = (0, 0, z_{uu})$$

$$\mathbf{x}_{uv} = (0, 0, z_{uv})$$

$$\mathbf{x}_{vv} = (0, 0, z_{vv})$$

The partial derivatives thus calculated are just in the right form required for curvature computation.

4.1.3 Examples

Based on the computed 3-D coordinates of the grid points, several surface patches of different types were analyzed using the differential geometry approach. On each of the surface patches, each quadrilateral of the stripe network was fitted by a B-spline surface using 4×4 neighboring points. Some 25 to 100 (u, v) change from 0 to 1 with a step 0.2 or 0.1) points interior to the quadrilateral on the fitting surface were picked, at which the normals and curvatures were computed.

- (1) The upper planar surface of the block in the jumble image.

Figure 4.2 shows the histograms of the number of surface points vs principal curvature. The units on the curvature axis are not the same for the principal curvatures and the Gaussian curvature. We put them in the same figure for simplicity. It is shown from the histograms that both the two principal curvatures as well as the Gaussian curvatures at most surface points are close to zero. Thus the surface patch is declared to be planar, since $K \approx 0$ and $\kappa_1, \kappa_2 \approx 0$.

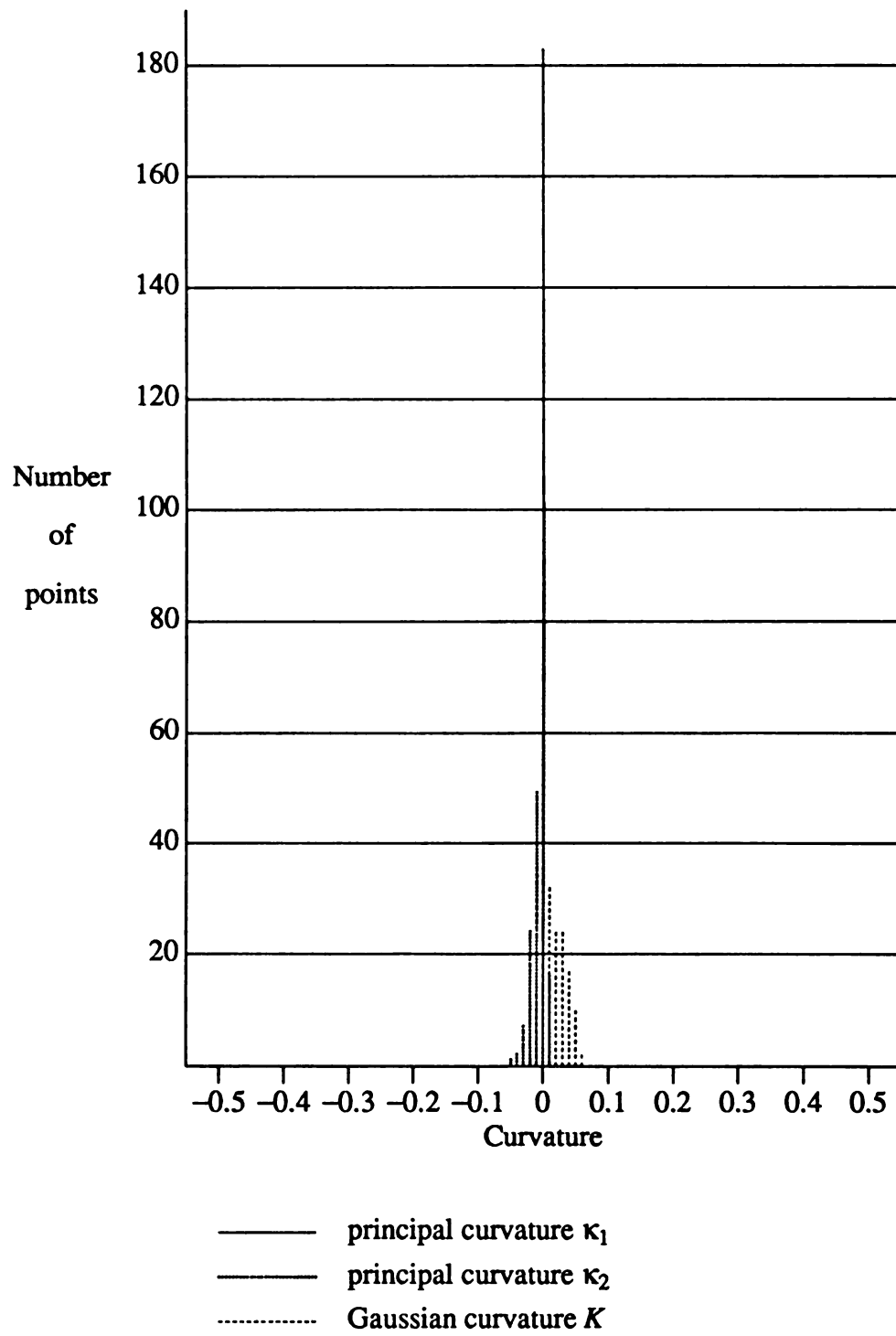


Figure 4.2 Histogram of curvatures of a planar surface patch

(2) Surface of the Coke can.

The histograms on κ_1 , κ_2 and the Gaussian curvature K are shown in Figure 4.3. The histograms show that the values of the two principal curvatures and the Gaussian curvature at various points on the surface patch are clustered with the cluster centers at $\kappa_1 \approx 0$, $\kappa_2 \approx -0.5$, $K \approx 0$. Accordingly, the surface is classified as parabolic. The non-zero principal curvature κ_2 is the normal curvature of surface curves in the direction that the values of curvature reach the maximum. Because the normal curvatures are the same almost everywhere (clustered at about -0.48 to -0.5, according to the histogram), these surface curves ought to be circles with a radius $r = \frac{1}{|\kappa_2|} \approx 2$. So, the surface is not only parabolic, but also cylindrical with radius 2 units. The unit here is the stripe spacing on the $Z=0$ plane, which is about 15mm. Hence the cylinder has radius about 30mm. As a matter of fact, the radius of the Coke can is indeed 32mm.

Corresponding to the principal curvature $\kappa_1 \approx 0$, the computed eigenvectors also show a cluster tendency with the cluster center being roughly the direction of the major axis of the cylinder.

(3) A potato in the 7-potatoes image

A first glance at Figure 4.4 shows that $K > 0$ and $\kappa_1, \kappa_2 < 0$. According to the classification theory, these curvatures define an elliptic surface patch. From the histograms, we see the two principal curvatures κ_1 and κ_2 being well separated, each spreads over a relatively wide range. We conclude that the surface curves downward in both principal directions ($\kappa_1 < 0$, $\kappa_2 < 0$), but more rapidly in one than in the other — a surface of an ellipsoid-like object, a reasonable description of a potato.

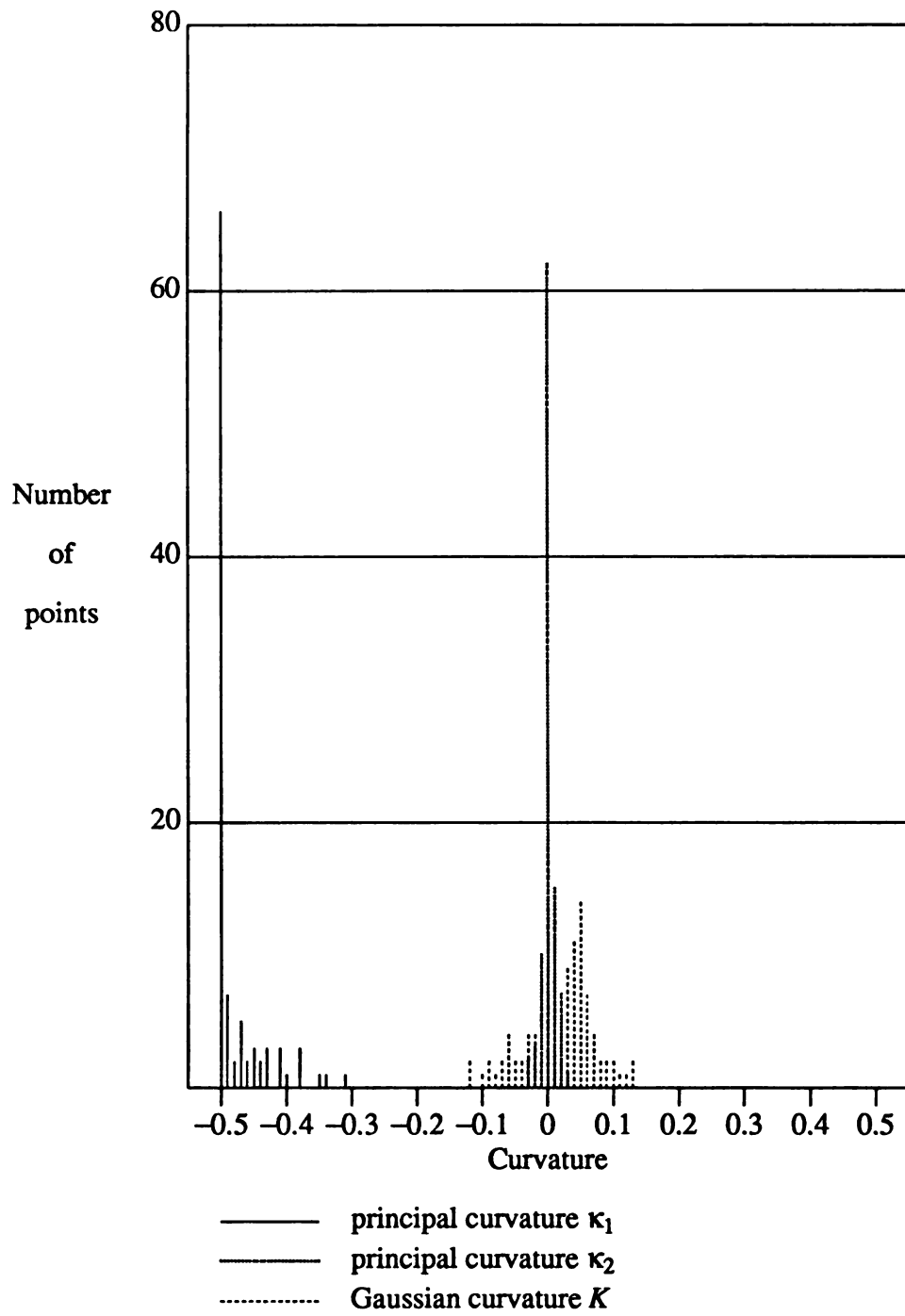


Figure 4.3 Histogram of curvatures on a cylindrical surface patch (Coke can)

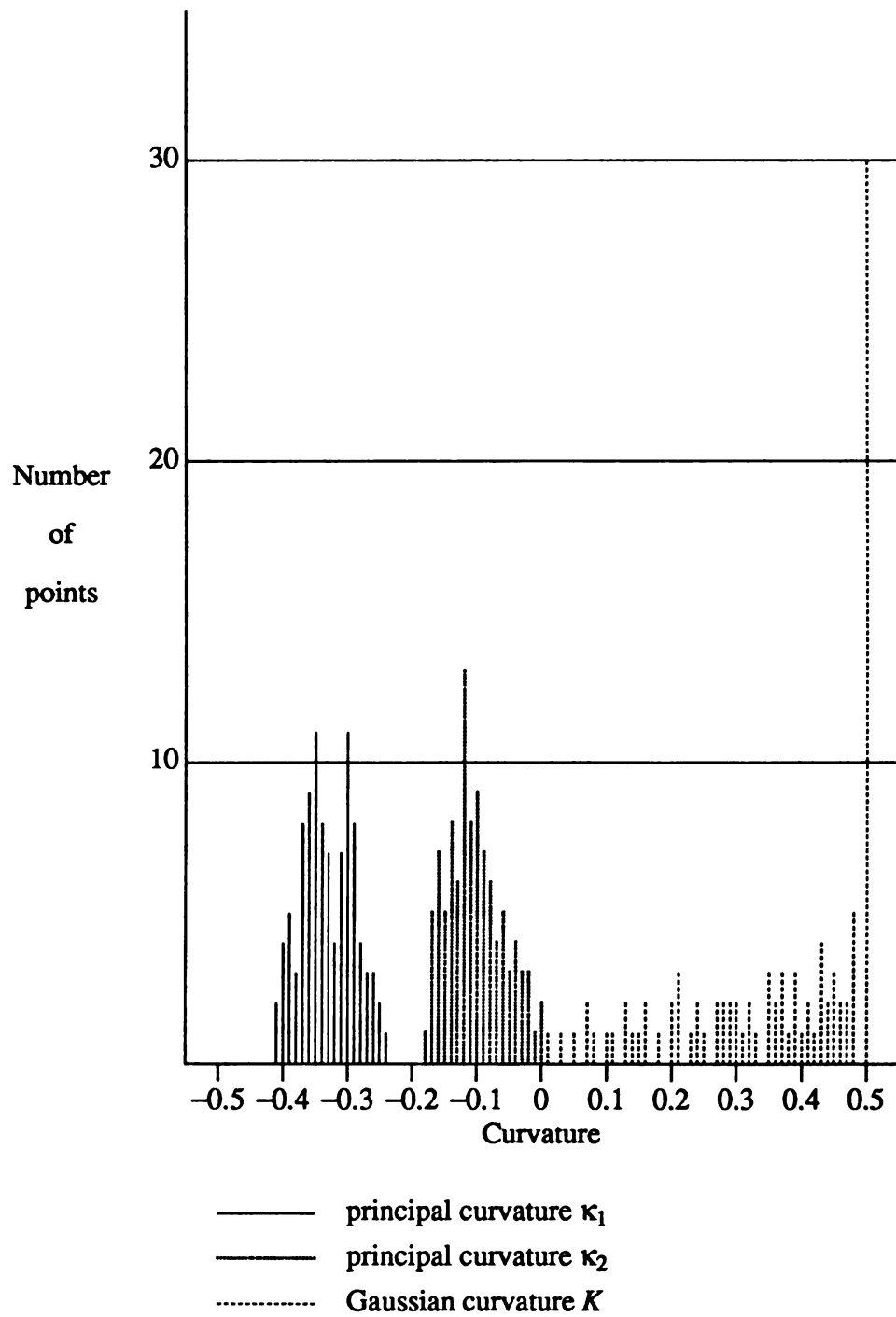


Figure 4.4 Histogram of curvatures on an elliptical surface patch (potato)

4.1.4 Occlusion relationships between virtual edges

The above curvature computation uses 3-D grid points interior to the stripe networks. But the stripe end-points are also very important in geometrical reasoning about the spatial relationships of the objects, especially about occlusions. If one surface is partially occluded by another surface with respect to the projector, the occluding surface will create a "shadow" on the occluded surface. In this case if a light stripe runs across the two surfaces, it will be interrupted at the place of the shadow. The 3-D data of the stripe end-points can be used to obtain this information. The computation is simple.

4.1.5 Summary

In this section, we have discussed an approach to inferring surface shape from the 3-D grid data. A stripe network is considered to indicate a surface patch, on which the 3-D data are available at equally-spaced grid points. Curvatures (κ_1 , κ_2 , K) at other surface points interior to the stripe network are computed based on a B-spline fitting to the sparse 3-D grid data. Clusters of the computed curvatures are sought, and surface shape is then classified using the curvature of the cluster centers. This technique is satisfactory for those objects that are of regular shapes, e.g. planar, cylindrical, spherical, etc. This work just shows how techniques used on dense range data [Ittner and A. K. Jain 85, Yang and Kak 86] can be applied to sparse 3-D data from stripes. Two problems remain : (1) What can be said if the curvatures are scattered all over the place with no cluster tendency? Are they of "arbitrary" shapes? (2) What about the points at the boundary areas of a surface patch: the above computation is valid only for the quadrilaterals *interior* to the stripe network? The first problem is indeed not easy to solve, since the differential geometry theory does not provide classification of surfaces that have nondeterministic curvatures. In many cases the surface has to be looked at locally, one small piece at a time. But, for certain types of surfaces, such as sine wave, the spread curvature measures may be useful for surface shape inference. The second problem requires more information about the

surface patches at the "critical" places — edges and boundaries. It is natural to probe other sources for more information. Stripe texture is one of the sources.

4.2 Surface Shape from Stripe Texture

So far we have computed surface solutions (Chapter 3) which are in fact locations of some isolated points on the surface. Also, we have done some surface shape inference based on the computed 3-D points (Section 4.1). Now, let's totally forget the 3-D points for a while and concentrate on what information we can pull out from the 2-D stripes.

The stripes seen by the camera weave stripe networks in the image. This kind of wire-frame network, or stripe texture, is often seen in computer graphics for 3-D display. Figure 4.5 shows such a stripe-network image. In this image only the bright stripes provide information about the objects, while other parts of the image, objects or background, provide little information, if any — they are homogeneously black. Through light stripes in the flat image, human eyes perceive something three-dimensional. We interpret 2-D stripes three-dimensionally because we have the built-in knowledge to reason about the 3-D scenes from 2-dimensional retinal images. For example, we know that a curved contour in the image never comes from a straight line in 3-D. Our task is to develop this kind of knowledge in our machine vision system and use it to interpret the images.

4.2.1 Relations of features in 2-D and in 3-D

Binford and Lowe [Lowe 84, Lowe and Binford 85] listed a set of ten relations of 2-D features and their corresponding 3-D inferences. The inferences are derived from very general assumptions about the image formation process, in particular that the camera and the light-source positions are independent of the objects in the scene. Under these assumptions, many types of alignments in an image are unlikely to arise by accident. More intuitively, the assumption is analogous to so-called the *general position* assumption that the image is taken from a representative viewpoint; i.e. a slight displacement of the objects in the scene would lead to essentially (qualitatively) the same image. Accidental alignment refers to the cases where certain image features arise by placing the

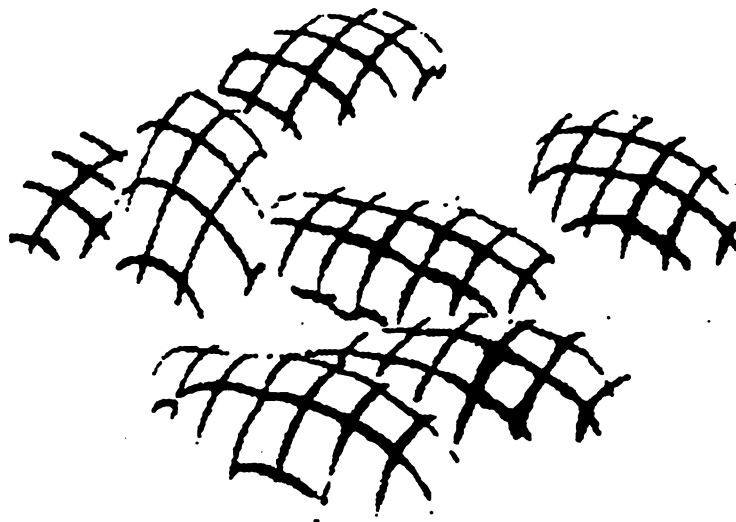


Figure 4.5 Light stripe networks in a striped image

camera in a special attitude. For example, a straight line segment in the image arises from a straight line in 3-D (general position), or the camera is viewing a planar curve in a direction that is in the same plane (accident alignment).

To infer three-dimensional shapes from the light stripe textures, which are 2-D features, a set of inference rules similar to that of Lowe and Binford's must be established. The following are the inference rules that I used in stripe texture analysis, under the general position assumption.

- (1) *A straight line in the image is cast by a straight line in 3-D. A curved stripe in the image is originated from a curved stripe in 3-D, and the 3-D curved stripe is on a curved surface.* In general, a curve in 3-D may not be on a curved surface, it may be a planar curve. But for light stripes, they are intersections of projected light sheets (3-D planes) and object surfaces. If a surface is planar,

the 3-D stripes on that surface must be straight. So a curved stripe in an image indicates a curve in 3-D which in turn indicates the surface on which the curved stripe lies is a curved surface. Note that a straight 3-D line does not imply a planar surface; it may be a line of least curvature on a cylindrical surface, for example.

- (2) *A continuous 2-D stripe indicates a continuous 3-D stripe* . Two disconnected 3-D stripes will be seen connected in the image (look like one single stripe) only when an accidental viewing position is taken to view the two 3-D stripes, which violates our "general position" assumption.
- (3) *Intersecting stripes (stripe network) in an image correspond to 3-D stripes on the same object surface*. By rule (2) each 2-D stripe corresponds to a 3-D stripe on an object surface, a 2-D stripe network (continuous stripes) indicates a single surface. But, because a camera viewing position not "accidental" to some stripes in the network is not necessarily a general position to other stripes of the same network, alignment of stripes belonging to different surfaces may occur. Although infrequent, this type of accident may be observed in an image, especially when the 2-D stripes are thick due to blooming. Accidental stripe connections in an image can be broken using some other information, such as finding no network-consistent surface solution for a false stripe network.
- (4) *Convexity and concavity of object surfaces can be determined by the 2-D stripe networks*. From the spatial relation between the camera, the projector and the scene in our experiments, we know that an image stripe curving upward (downward) toward its ends indicates a concave (convex) 3-D stripe on a surface. "Up" and "down" were defined in Section 3.1.2. Convexity and concavity of the object surfaces are then determined by the 3-D stripes.

4.2.2 Stripe texture

If the camera were placed at exactly the same position as the projector, the light stripes in the image would be a regular grid (perhaps with a little lens distortion), and we would not be able to extract any useful information. In this case all the stripes on the object surfaces are accidentally aligned with the viewing direction — seen as straight lines in the image. By increasing the displacement of the camera and the projector, the light stripes seen in the image are distorted as a function of the object surfaces. It is the distortion of the stripes that provide information we may use to discover the inverse function, i.e. to recover the object surfaces. Stripe textures in an image, which are distorted grids of tessellations, are useful for analyzing surface characteristics. Two kinds of texture primitives are available : 4-sided tessellar cells and stripe curves.

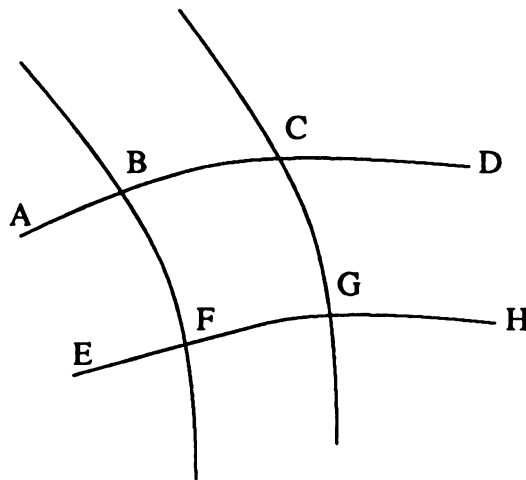


Figure 4.6 Stripe texture primitives

Tessellar cells

A tessellar cell is a tile in a stripe network, composed of four vertices and four edges of two adjacent X-stripes and two Y-stripes. In Figure 4.6, BCGF is a tessellar cell. A set of features are defined on the tessellar cell primitive.

- (1) Standard deviation of side length, σ_l . This feature measures the degree of distortion of the cell from a rhombus shape.
- (2) Differences of two opposite sides, denoted by Δl_1 and Δl_2 . A cell on a planar surface tends to have equal lengths of opposite sides, but one from a curved surface does not.
- (3) Standard deviation of angle, μ_α . This feature also measures the degree of distortion.
- (4) Differences of two opposite angles, $\Delta\alpha_1$ and $\Delta\alpha_2$. The differences will be close to zero if the cell is on a planar surface. They will be significantly different from zero if the tessellar cell is on a non-planar surface.
- (5) Average k-curvature of sides of the cell, μ_c . The k-curvature of a side is defined as the mean of k-curvatures at each point on the side, where the k-curvature at a point p is defined as the angle change between line ap and pb , where a and b are k pixels away from p in opposite directions ($k = 5$ in our experiments) [5]. Note that the measured "curvature" here is not the same as that in section 4.1. Here the "curvature" is an approximation of the curvature of a planar curve (in the image plane), but in section 4.1 we were talking about curvatures of 3-D surfaces.
- (6) Compactness, defined as p^2/A , where p is the perimeter of the cell and A is its area.

Stripe curves

A stripe curve in an image is the image of the projection of a single grid line. ABCD in Figure 4.6 is an example of a stripe curve. Stripe curves reveal planarity of object surfaces, since 2D straight lines generally imply 3D straight lines, as in the inference rule (1) of section 4.2.1. Features of a stripe curve primitive include :

- (1) Average value of curvature, μ_{cur} , and standard deviation of curvature, σ_{cur} . Curvature is defined as the angle between two consecutive stripe segments on the curve.
- (2) Average difference of lengths of consecutive stripe segments on the stripe curve, $\mu(\Delta_d)$. If a stripe is a straight line, its vertices are nearly equally spaced, thus the differences of lengths are close to zero. Here, we assume the viewing angle effect on the spacing is small.

Stripe curves are detected based on the 2D grid networks and certain topological constraints. Refer to [Stockman and Hu 85] for details.

4.2.3 Stripe texture classification

Texture classification is done in a hierarchical fashion as (1) planar vs. nonplanar, (2) convex vs. concave vs. irregular, and (3) spherical vs. cylindrical.

Planar surfaces are distinguished from nonplanar ones based on features of tessellar cells and stripe curves. A 1-NN classifier is designed to identify planar and nonplanar cells based on the six features defined earlier. Euclidean distance in the feature space is used to determine near neighbors. Notice that this method only classifies individual tessellar cells, not the texture itself. But, a texture may indicate a planar (nonplanar) surface if a majority of its cells are classified as planar (nonplanar). This can be used as supplementary evidence to decisions based on properties of stripe curves.

Histograms of μ_{cur} and $\mu(\Delta_d)$ of stripe curves are constructed. Since stripes on a plane are always straight, their curvatures are clustered at zero. And, since equally spaced grid points produce nearly equally spaced stripes on planar surfaces, $\mu(\Delta_d)$'s are also concentrated at zero. But stripe curves on nonplanar surfaces have nonzero curvatures, and are not equally spaced in general. Thus, a peak near zero in each of the feature histograms indicates straight stripes. If the straight stripes constitute an absolute majority of stripes of a texture, then that texture is assumed to indicate a planar surface.

Once a texture is determined to be nonplanar, we come to the next layer in the classification hierarchy — determining if it represents a convex or concave surface. This is done via the average curvature (μ_{cur}) and standard deviation of curvature (σ_{cur}) of the stripe curves. For nonplanar surface patches, μ_{cur} and σ_{cur} of X-stripes and Y-stripes are computed separately. From the inference rule (4) of section 4.2.4, we have the following criteria for determining convexity and concavity of a surface patch :

$$(X_{\mu_{cur}} < 0) \text{ and } (Y_{\mu_{cur}} \geq 0) \Rightarrow \textit{convex}$$

$$(X_{\mu_{cur}} > 0) \text{ and } (Y_{\mu_{cur}} \leq 0) \Rightarrow \textit{concave}$$

where each boolean subexpression is based on the majority of stripe curves in that direction. If the above two conditions are not satisfied then the surface patch is said to be of irregular shape (neither convex nor concave). The value of σ_{cur} is also used to adjust the decision made above. If a surface has been classified as convex (concave) but the stripe curves on the surface have large σ_{cur} 's, the surface is reclassified as irregular although it is basically convex (concave). Note that a convex or concave surface (cylindrical, for example) has stripe curves with near-zero σ_{cur} .

The texture gradient based on tessellar cell primitives is also useful in distinguishing cylindrical and spherical surfaces. We compute gradients along four directions (X-stripe direction, Y-stripe direction, and two diagonal directions) on four features (σ_l , σ_α , $\Delta\alpha_2$, and shape). A cylindrical surface would have a value relatively small in one

direction (along its principal axis) but spherical surfaces have relatively large values in all four directions.

4.2.4 Experimental Results

Fifteen striped images were analyzed containing 27 surface patches, including 10 planar surfaces, 6 convex cylindrical surfaces, 5 convex spherical surfaces, 3 concave spherical surfaces, and 3 irregular shaped surfaces. The objects were viewed in a variety of orientations. There were 792 tessellar cells and 332 stripe curves in total.

For the 1-NN classifier, 106 randomly selected cells from the total population of 792 were in the training set. The remaining 686 cells formed the testing set. The results are listed in Table 4.1. Using a majority decision rule on individual cells, 2 out of 27 surface patches were misclassified (which had a small number of cells) and another two patches remained undecided (planar-nonplanar tie).

Table 4.1. Surface patches, texture primitives, and classification decisions.

No.	surface patch	# of tessellar cells in testing set	decision using tessellar cell features	# of stripe curves	decision using stripe curve features
1	jar	35	N	17	convex
2	ball	64	N	22	convex
3	book	48	P	15	planar
4	notebook	28	P	17	planar
5	roll of paper	35	N	22	convex
6	board eraser	3	P	3	convex *
7	metal block	8	P	8	planar
8	notebook	6	P	7	planar
9	board eraser	3	N †	4	planar
10	piece of wood	14	- ‡	11	planar
11	book	9	N †	9	planar
12	block	14	- ‡	11	planar
13	bowl	28	N	14	concave
14	worktable	27	P	13	planar
15	cylinder	13	N	10	convex
16	cylinder	11	N	9	convex
17	roll of paper	45	N	9	convex
18	grapefruit	13	N	11	convex
19	bowl	26	N	15	concave
20	bowl	18	N	14	concave
21	apple	5	N	6	convex
22	apple	3	N	6	convex
23	tennis ball	5	N	6	convex
24	waved paper	21	N	18	irregular
25	waved paper	22	N	20	irregular
26	cloth	59	N	19	irregular
27	garbage can	123	N	30	planar *

P — planar; N — nonplanar.

† — misclassification using the majority decision rule.

‡ — undecided using the majority decision rule (P-N tie).

* — misclassification using stripe curve features.

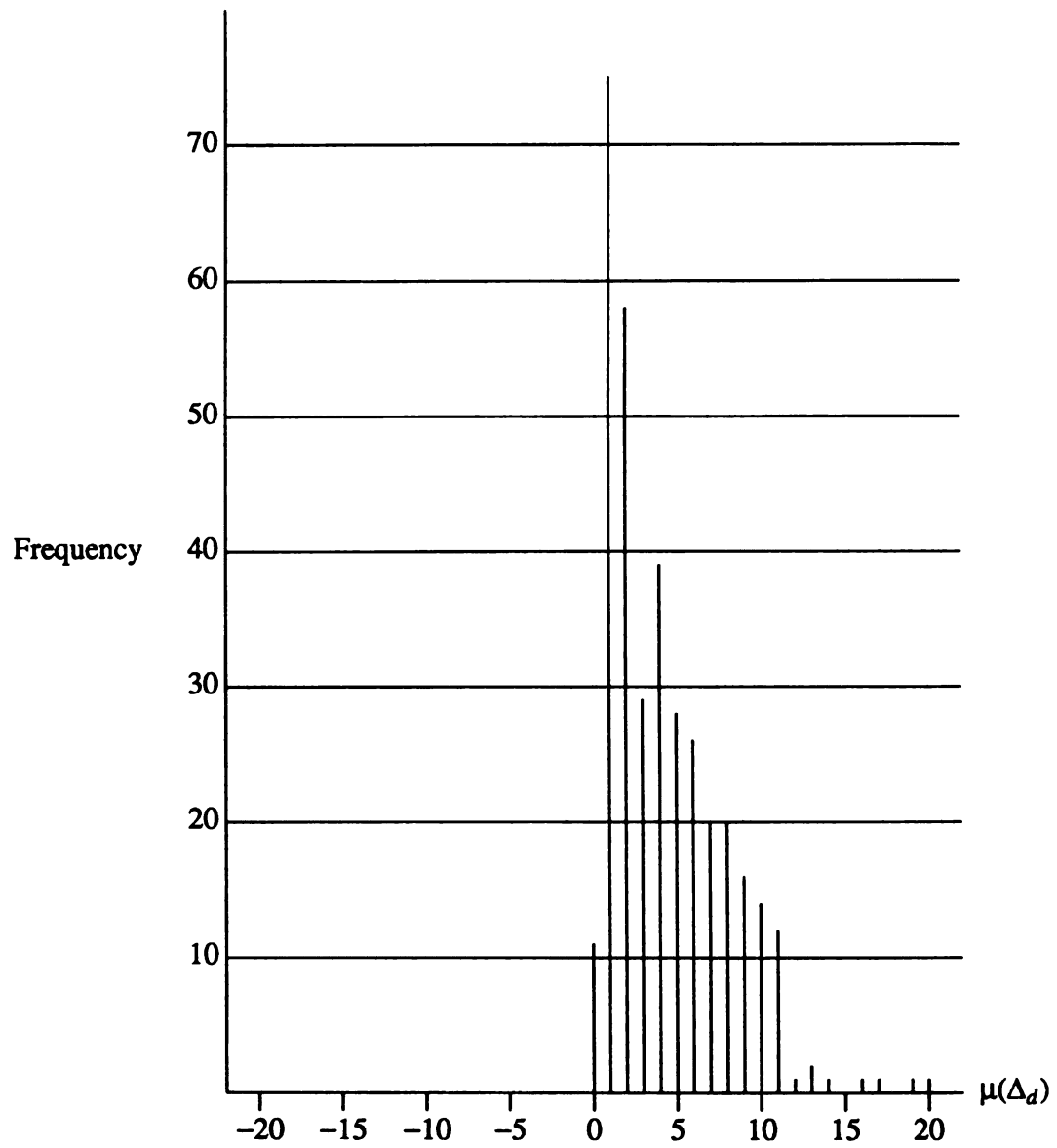


Figure 4.7 Histogram on $\mu(\Delta_d)$ of stripe curves

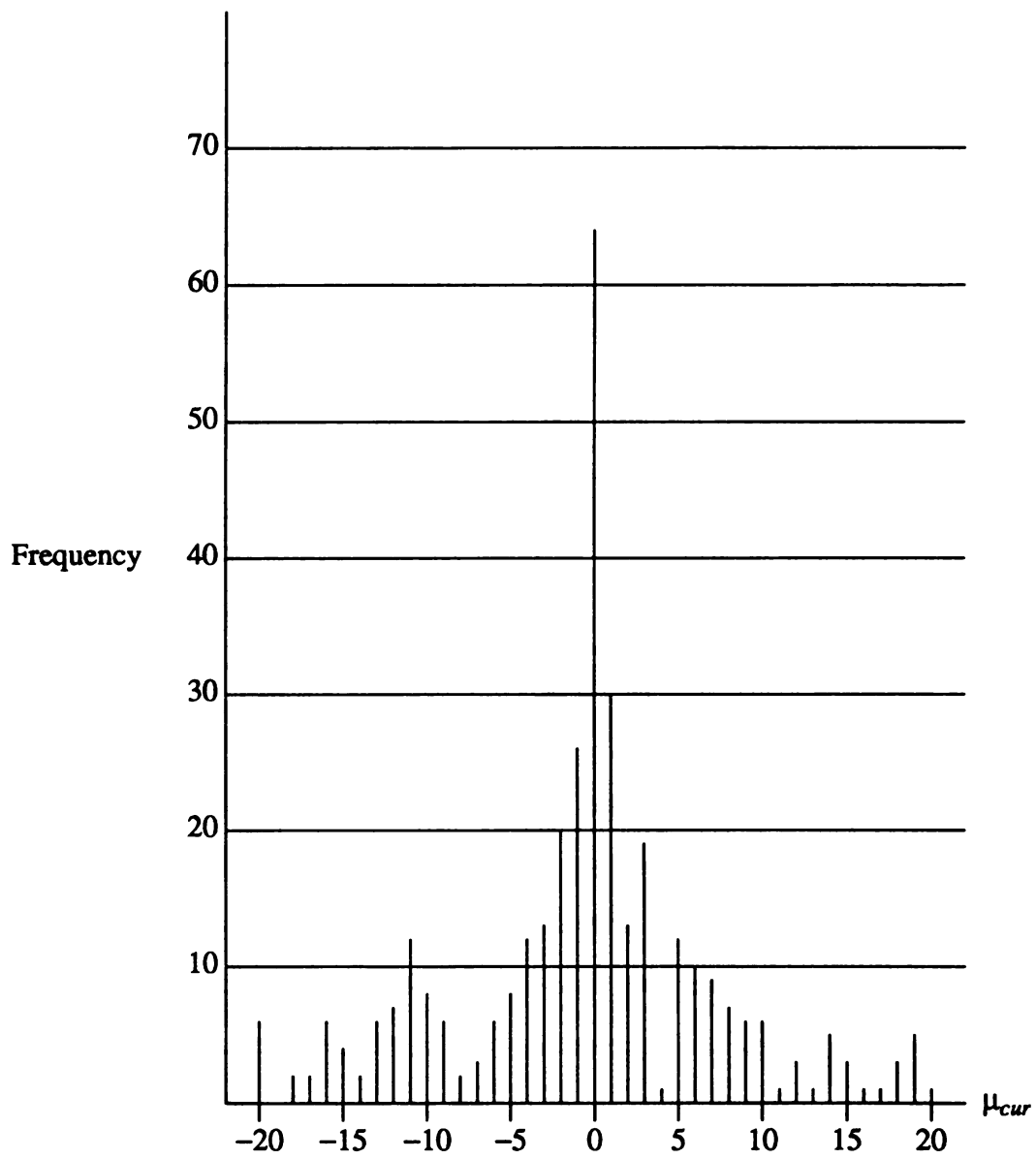


Figure 4.8 Histogram on μ_{cur} of stripe curves

Histograms of two stripe features, $\mu(\Delta_d)$ and μ_{cur} , were constructed based on data from all 27 surface patches as shown in Figure 4.7 and Figure 4.8. We observe that there is a peak near zero in both histograms, which indicates presence of planar textures. In the histogram of curvature, two valleys were found around the peak. Using these valleys (marked by arrows) as thresholds, the stripe curves were well classified as planar or non-planar. For the nonplanar textures, concavity and convexity were determined using the algorithm described in section 4.2.3. The result is shown in Table 4.1. Among the 27 surface patches only 2 (surfaces #6 and #27) were misclassified. The misclassification was due to (1) too few texture primitives (#6), and (2) similarity between a planar surface and a cylindrical surface with large radius (#27).

To distinguish cylindrical and spherical surfaces, a gradient operator was applied on the textures. Since all tessellar cells were organized according to their spatial relationships and a texture was represented as a two-dimensional array in the detection phase, texture gradient is computed the same way as on a gray tone image. Gradients along four directions (X-stripe direction, Y-stripe direction, and two diagonal directions) were computed. This gradient operation was performed on surfaces that have been determined to be either convex or concave. Results on 7 such surfaces are listed in Table 4.2. We can see from Table 4.2 that cylindrical surfaces have average gradient magnitudes relatively small in one of the four directions (indicated by *). That direction gives roughly the orientation of the principal axis of a cylindrical surface.

Table 4.2. Average gradient values on 4 features along 4 directions for 7 surface patches. Cylindrical surfaces have values small in one of the 4 directions (indicated by *) that are assumed to be their principal axes. Surface patch numbers refer to Table 4.1.

surface patch	gradient direction	features			
		compactness	σ_l	σ_α	$\Delta\alpha_2$
1. cylinder	X	0.77	2.91	6.36	8.40
	Y	0.83	3.19	5.88	9.67
	diagonal 1	0.92	4.06	7.62	7.25
	*diagonal 2	0.72	2.37	4.88	5.42
2. sphere	X	1.39	1.84	7.43	7.69
	Y	1.07	1.98	4.64	7.32
	diagonal 1	1.92	1.63	11.55	3.94
	diagonal 2	1.01	3.21	4.39	4.53
5. cylinder	X	0.60	2.88	4.49	7.99
	Y	0.71	2.54	3.93	7.08
	*diagonal 1	0.51	0.60	3.12	3.63
	diagonal 2	0.80	6.81	6.95	10.05
13. sphere	X	0.93	2.38	6.36	7.48
	Y	0.89	2.35	6.35	7.17
	diagonal 1	0.64	3.44	3.36	8.24
	diagonal 2	1.02	3.18	8.37	6.66
16. cylinder	* X	0.15	0.52	1.25	1.37
	Y	1.83	2.87	15.17	1.97
	diagonal 1	1.78	2.50	16.76	1.91
	diagonal 2	1.81	3.21	16.27	1.54
17. cylinder	X	0.95	3.30	11.09	2.95
	* Y	0.26	0.60	1.11	3.00
	diagonal 1	1.00	3.38	11.74	2.38
	diagonal 2	0.92	3.01	10.96	2.02
18. sphere	X	0.64	2.16	10.40	5.87
	Y	0.68	2.49	10.24	6.00
	diagonal 1	1.05	3.86	6.39	7.75
	diagonal 2	0.64	1.40	14.11	8.39

4.3 Virtual Edges

In addition to the surface shapes and locations of objects in the 3-space, the spatial relationships among the objects are also important to scene understanding. One source from which the occluding-shadow relations with respect to the projector may be derived is the set of virtual edges [Marr 82]. The *virtual edges* in striped images are the "edges" that form the boundaries of the stripe networks. The network boundaries are constructed by the "turning right" algorithm in Section 3.4.5 using the stripe end-points.

The geometric constraint (G4) of Section 3.3.2 says "If two stripe end-points lie on the same projector ray in 3-D then their stripe numbers are the same and one lies on a surface between the other and the projector". By this constraint we can find the occluding-shadow relations of surfaces with respect to the projector by looking at the relations between stripe end-points of same stripe number to see if they lie on the same projector ray. The algorithm is summarized as follows.

Algorithm (virtual edges). For each pair of stripe networks n_1 and n_2 , find a stripe pair (s_1, s_2) such that they have the same stripe number (x_g or y_g), and $s_1 \in n_1, s_2 \in n_2$. From the 3-D coordinates (x_w, y_w, z_w) of the end-points of the two stripes we can use the projector calibration matrix M_P to find their projections (x_g, y_g) on the slide plane. If end-point $e_1 \in s_1$ and $e_2 \in s_2$ are very close in the slide plane, we know that they lie on the same projector ray. The one that is closer to the projector (can be determined by their 3-D positions) *occludes* the other. The above procedure is applied on all stripe pairs of n_1 and n_2 . After occluding-shadow relations of individual stripe end-points are found, the virtual edge relations can be established since a virtual edge is simply a sequence of individual stripe end-points, with well defined order.

Refer to the jumble image we experienced in Chapter 3 whose stripe networks are in Figure 3.10, the virtual edge algorithm resulted in Table 4.3.

Table 4.3 Shadow and Shadow-making virtual edges with respect to the projector

Occluding point			Shadow point		
id	x_g	y_g	id	x_g	y_g
122	11.00	4.71	141	11.00	4.73
199	11.95	2.04	132	11.95	2.05
199	11.95	2.04	132	11.95	2.05
116	11.49	3.00	134	11.61	3.00
121	11.08	4.00	139	11.18	4.00
123	10.63	5.00	140	10.71	5.00
124	10.20	6.00	149	10.32	6.00
37	10.89	7.11	148	11.00	6.64

Notice that point 199 and 132 appear twice in the table because both are end-points of an X-stripe and end-points of a Y-stripe. For point 37 and 148, their occluding-shadow relation bears low confidence, since their y_g 's are not very close. For virtual edges, the order in which the end-points are extracted is 199, 116, 121, 122, 123, 124 for the upper network, and 132, 134, 139, 141, 140, 149 for the lower network. It is seen that there is a perfect match between the order of the end-points in the two networks and the occluding-shadow relationship. Hence we have established the spatial relation between

the two surface patches in terms of their relative positions with respect to the projector. In general there may not be a perfect match, but the virtual edge relations can still be established if most end-points on the two virtual edges have occluding-shadow relations.

4.4 Discussion and Conclusions

In this chapter we have discussed inference of surface shapes from light striping through curvature computation (3-D) and stripe textures (2-D). The results obtained are rough shape descriptions, both qualitative as planar, concave, convex, etc., and quantitative as 3-D locations of the boundary points of stripe networks, radii of spherical or cylindrical surfaces, and so forth. Experiments showed that surface shapes of certain smoothly curved objects could be deduced reasonably well.

Using the computed 3-D points to fit a surface patch and find local curvatures at various points on the fitted surface patch to infer shapes is a standard way to analyze a surface [Ittner and A. K. Jain 85, Besl and R. Jain 85, Medioni and Nevatia 84, Laffey, Haralick and Watson 82, Ponce and Brady 85, Yang and Kak 86, Besl and R. Jain 86]. But in some cases we need to analyze surface shapes using only 2-D information (3-D data are not available, for example); and in some other cases 2-D information is itself adequate. Stripe patterns in the image, or stripe textures, are a good source from which the desired information can be obtained. In a 2-D image, the stripe textures provide information about the distortion of the stripes, which is a function of the object surfaces. The experiments on stripe texture analysis showed very convincingly that the 2-D stripes may provide essential information about object shapes. On the other hand, 2-D stripes reveal surface shapes only qualitatively, such as planar, convex, etc., not quantitatively. Hence both 3-D and 2-D inferences are important and useful.

Related work [Shrikhande and Stockman 87] showed that surface normals at stripe grid points can be directly derived with good accuracy without computing locations of 3-D surface points. This information is also useful and can be combined with the fitted

surface in shape inference.

In shape inference using 3-D data and 2-D stripe textures, several issues remain to which special attention must be paid :

- (1) Gaussian curvature is computed only for points interior to stripe networks. This is because surface fitting itself, as any other kind of data fitting, is not good for points at the boundaries. To solve this problem, we need methods specially designed for handling boundaries. Fusion of light striping with intensity is one of the methods, which will be the topic in Chapter 5.
- (2) Qualitative description of surface shape is made locally unless the surface is regular of some sort. An arbitrarily curved surface should be broken down into pieces, each having its own "global" description.
- (3) Surface patches must be "large" enough in order to be described through stripes. As stated in Chapter 2, this is one of our basic assumptions about surfaces. If a surface patch has too few stripes it will not be described properly, or will simply be ignored by the processing procedures.

CHAPTER 5

Multi-channel Vision—

Fusion of Light Striping and Intensity

It is seen from the previous chapters that structured lighting technique can be effectively used for 3-D sensing and shape inference. But it is also clear that using structured lighting alone we may not be able to obtain all information necessary for developing a proper object representation for recognition. An obvious example is an object surface that is visible to the sensor (camera) but invisible to the projector. There are no light stripes shining on such a surface and hence no information about it is available. A natural way to overcome this difficulty is to look for information from other sources, such as intensity images of the same scene. Multi-channel information is then fused together to yield a description of the scene. A fusion scheme is presented in this chapter that combines information from light striping and gray tone image to develop symbolic descriptions for the surface patches and intensity edges according to some "inference rules". The symbolic descriptions may be combined in the higher level analysis to segment the image into objects.

5.1 Object Description

The goal of 3-D scene analysis is to recognize the objects in the scene. In order to accomplish this task, object models which are the descriptions of the physical objects, must be available somewhere in the system. The extracted features from the images should also be described using the same language as used in the models; — the representation of the sensed structure can be matched against the representation of the models for recognition. As stated in Chapter 2, we use the surface-boundary representation that consists of the following basic elements (repeat of Ch 2).

1. surfaces

- (a) qualitative — type (planar, convex, concave, cylindrical, etc.),
- (b) quantitative — boundary, 3-D location, orientation;

2. edges

- (a) qualitative — type (extremum, blade, fold, mark, shadow),
- (b) quantitative — location, length (in chain-code form);

3. surface-surface relationships

- (a) occlusion,
- (b) adjacency,
- (c) belonging to same or different objects;

4. surface-edge relationships

- (a) surface boundaries,
- (b) surface marks,
- (c) shadow edges,
- (d) intersection (common border) of two surfaces.

From the discussion in Chapter 3 and 4, we may find that light striping can provide elements 1 and 3(a) in the above representation. Now the question is, "Can other

elements of the representation be obtained from light striping ?"

5.2 Light striping is not enough

We have discussed the light striping technique in 3-D sensing and surface shape analysis. This technique has been shown to be promising. Besides providing locations of surface points in the 3-space, the sensed 3-D data have the following properties.

- (1) They are sparse compared with dense data obtained using direct range sensors. Because of the sparseness, the computational burden often inevitable in processing dense data is somewhat relieved.
- (2) Although sparse, they are *good* representatives of the dense 3-D population, because they cover the major part of visible surface due to the way the feature points are created. The spatial relationships between the 3-D points are well defined in terms of the stripes that connect these points. Hence, a good surface fitting may be constructed based on these representative 3-D points.
- (3) They are ambiguous in the sense that a surface patch may be at one of several locations in 3-space. But as far as surface shape is concerned, each of the several surface locations is equally good for surface shape inference.
- (4) They are available only in the areas visible to both the projector (there are stripes) and the camera (stripes appear in image). For areas where no stripe appears due to occlusion or shadowing, no information can be obtained.

Besides the weakness stated in (4) above, some problems introduced in 2-D processing may also arise. On a smooth surface the stripes would gradually fade out as the surface turns away from the viewer. These stripes may not reach the real object boundaries before they die out. A stripe extraction procedure (brightness thresholding, say) will fail to pick up the fading tails of the stripes, but the tails, especially the end-points of the tails, are indeed very important features for 3-D analysis. For the same reason that

brightness of stripe pixels varies as the surface orientation varies from point to point, global brightness thresholding may result in broken stripes and fragmented stripe networks, which introduces more difficulties in further processing steps. There seems to be no way to get around these problems using only light striping. We ought to resort to other sources for the missing information. Fortunately, an intensity image of the same scene viewed at the same position and in the same direction is easily obtained by simply switching off the projector. Exact registration of the two images (striped and intensity) is guaranteed since neither the camera nor the objects in the scene have been touched. The fusion of information from striped and intensity images will be the topic of this chapter.

5.3 Comparison of the two information channels

A striped image and its intensity counterpart each has its strengths and weaknesses. To better understand why the combination of information from these two different sources is advantageous, let's look at some examples. In Figure 5.1, a gray-scale image and its corresponding striped image are shown in (a) and (b), respectively. (c) is the result of a gradient operation on the gray-scale image followed by a thresholding, and the overlay of the edge map and striped regions is in (d). From the striped image alone, we can compute 3-D positions of the grid points, determine the rough surface shapes (planar in this case), detect blades and folds, and segment the image according to the stripe networks (two segments in the example). But it provides no information in the areas where light stripes are occluded, and no clues about whether the two surfaces belong to the same object. On the other hand, the gray-scale image may not catch discontinuities of surface normal (the concave edge is missing, for instance), and may have difficulties in segmenting surfaces. But it offers fine boundaries and texture. In the example, the gray-scale image suggests that the two striped surfaces belong to the same object because of well-connected outer boundaries. Also the clear edge of the triangular-shaped surface is useful in breaking down the larger striped surface into two sub-regions, which will be

confirmed by the bending of the stripes. In section 5.5, we will find that this particular edge is a *fold*, and the two sub-regions are two surfaces meeting at the fold.

A second example of multiple objects is shown in Figure 5.2. Notice that the striped regions do not exactly agree with the contours, due to the continuous change of surface normal and stripe fading. But they provide a good preliminary segmentation, which is not easy to get from the intensities alone. A more accurate segmentation can be obtained by extending the striped regions with the gradient contours as boundary conditions.

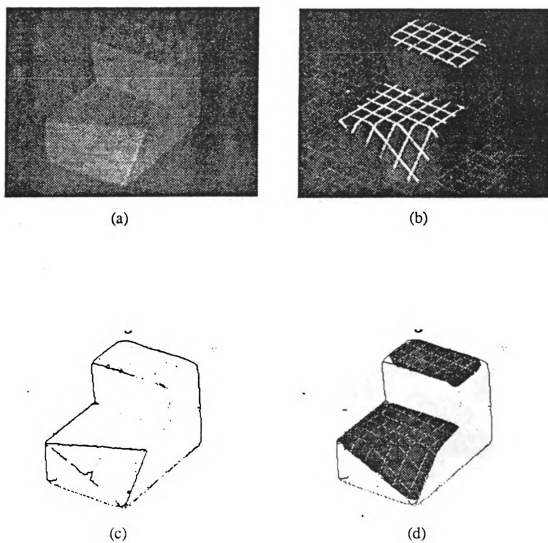


Figure 5.1 Fusion of intensity and light striping — example 1
(a) intensity image (b) striped image
(c) edge map (d) overlay of edges and striped regions

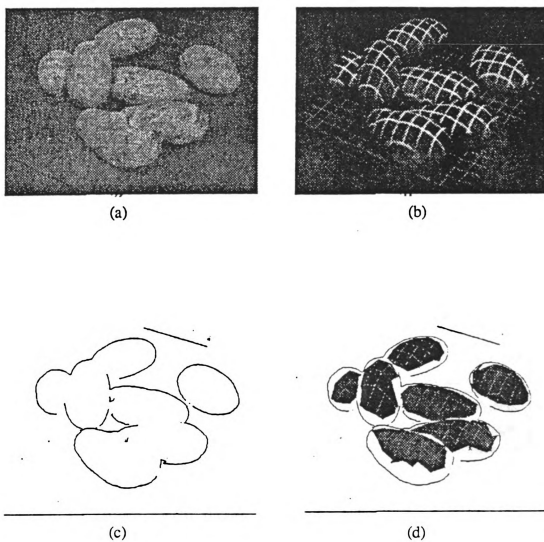


Figure 5.2 Fusion of intensity and light striping — example 2
(a) intensity image (b) striped image
(c) edge map (d) overlay of edges and striped regions

Table 5.1 gives a comparison of the two information channels. Combination of the two channels will certainly facilitate the strengths of each, and is necessary for generating an object representation that will be suitable for the next step of processing up the visual chain, e.g. model matching and recognition.

Table 5.1 Comparison of two information channels

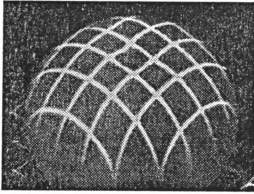
	Striped Image	Intensity Image
Information	gross surface shape, location, boundaries and segmentation; detect folds (crease edges) in surfaces and some blades & extrema	local surface albedo and orientation (under controlled illumination); little shadowing; fine boundaries and textures; can see into some of surfaces in shadow of projected light
Factored Out	albedo; fine texture	no direct 3-D info (no knowledge of projector)
Problems	shadows; occasional accidents; some ambiguity in 3-D solution; will miss illumination or albedo discontinuities	segmentation into objects; difficult to get gross shape; can miss discontinuities of surface normal (folds)

5.4 Stripe extension

Since light stripes in an image are much brighter than other part of the image, a global thresholding would be a proper way to extract the stripes. But on the other hand, because the intensity received in an image is a decreasing function of the angle between the viewing direction and the surface normal, the intensity value on a light stripe that runs across a smooth curved surface will fade out as the surface turns away from the camera (or projector), and will vanish when the surface normal is at 90° to the imaging direction (projection direction). Hence, a global thresholding may cut off some stripes in the fading areas. To recover the thresholded part of stripes, a local thresholding may be useful.

Local thresholding

Figure 5.3 (a) is a striped image of an orange, which is part of the image in Figure 2.2 (c). Figure 5.3 (b) is the result of a global thresholding on (a). It is seen that the tails of some stripes were cut off, with detected stripe end-points at somewhere toward the middle of the stripes. To get better end-points, we apply local thresholding on the regions that are in the vicinity of the detected stripe end-points. To find a proper threshold value for thresholding a subimage, the common approach is to construct the histogram of intensity in the subimage, and find a valley in the histogram. But, identifying valleys in a histogram is itself a non-trivial problem when there are no clearly defined peaks and valleys, which is the case in our situation where the intensities of both stripe pixels and background pixels scatter over a wide range. Instead of working on histograms, we adopted a heuristic approach. The heuristic is that the stripe tails, although weak in intensity, are in general still brighter than the near-by background pixels, and the majority pixels in the local region belong to the background. That is to say, the stripe pixels occupy a certain portion at the high end of the intensity range. We need to locate a cut point at the right place. Two methods can be used to determine the cut point.



(a)



(b)

Figure 5.3 (a) striped image of an orange (b) result of global thresholding

method A. Suppose the local region is an $n \times n$ subimage. The stripe tail may run across the region so it will be n pixels long; or it may terminate in the middle of the region and hence will be shorter. For a stripe tail 3 pixels width, it has at most $3n$ pixels; that is $\frac{3n}{n^2} = \frac{3}{n}$ of the region. For $n = 15$, for example, no more than the top 20% high intensity pixels may be on the stripe. The results of thresholding using 15% as the cut point is shown in Figure 5.4 (a).

method B. Because most pixels in the local area are background ones, the average intensity would be closer to that of the background than to the stripe. Those pixels having higher-than-average intensities that deviate very much from the mean are classified as stripe pixels. Using 9σ as the threshold, where σ is the standard deviation of intensity, the result is shown in Figure 5.4 (b).

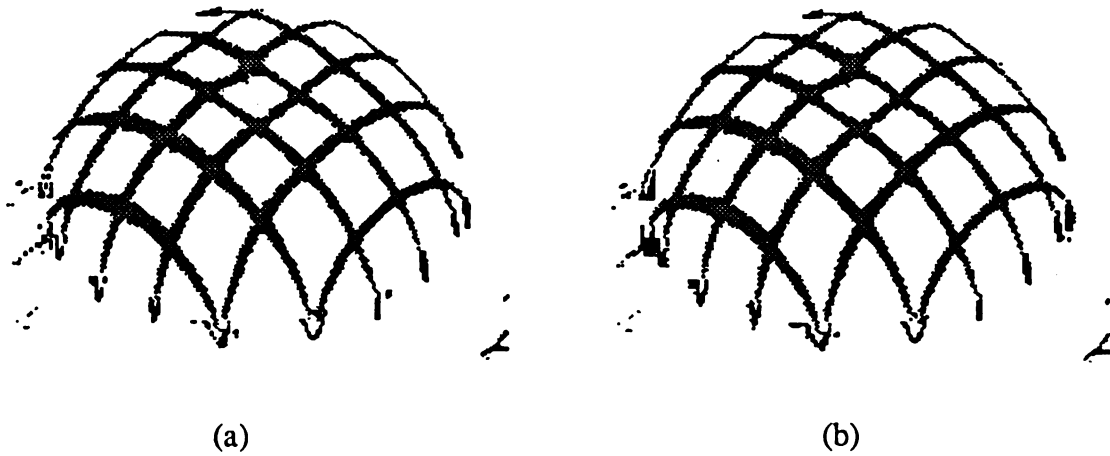


Figure 5.4 Local thresholding results using threshold value at
(a) top 15% intensity
(b) 9σ (σ is the standard deviation of intensity)

Notice that there may be some other stripes (besides the one we are working on) running into the local region. We don't want those bright stripes to interfere with the processing. A simple way to avoid this is to consider not all pixels in the region but only those whose intensities are *lower than or equal to* that of the detected end-point p , and ignore the higher-intensity pixels because they have already been taken care of by the global thresholding.

Although local thresholding can to some extent recover the faded stripe tails, which are important in relating surface patches (indicated by stripes) and boundaries, it may not be applicable in general, because the choice of the parameters (size of local region, threshold values etc.) is ad hoc, and, a lot of post-processing work needed to relate the locally thresholded regions to the original stripes may be too complicated and not very reliable.

Stripe extension

Local thresholding may improve the accuracy in locating the real stripe end-points, which are important for determining surface boundaries. But often the boundaries of the striped regions do not agree with the boundaries of object surfaces, as shown in the potato image in Figure 5.2 (d). The striped regions give an initial segmentation; each of the regions corresponds to a surface. On the other hand, the contours such as the one in Figure 5.2 (c), represent surface boundaries but are not easy to segment, because the contour image may still be quite noisy and incomplete. This suggests that a better segmentation may be achieved by "growing" the initial striped segments using the contours as termination condition. One way to do this is to extend the stripes to the contours.

For each stripe we can fit a curve using the grid points on the stripe. Points external to each end of the stripe are calculated using the fitted curve, or we simply linearly extend the stripe, until one of the following occurs.

- (1) a gradient contour is encountered.
- (2) A pixel in some other segment is hit.
- (3) Extension runs into a place that is likely to belong to the background (very low intensity, say). This condition is necessary since gradient points usually do not form closed contours; stripe extension may escape through the gaps between gradient contours.

Figure 5.5 illustrates the effect of stripe extension. (a) is the initial striped segments of the potato image along with the contour map obtained by gradient operation. Actually in this example, the gradient operation was not applied directly to the original intensity image; rather the intensity image was preprocessed by a logarithm operation that rescaled the intensities according to an appropriate logarithm curve. In (b) the stripes are extended, and (c) shows the resulting segments.

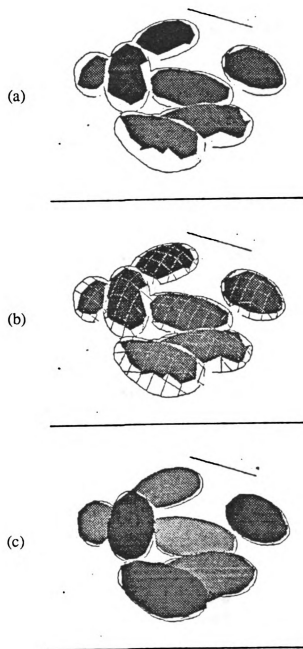


Figure 5.5 Segmentation using both stripes and intensity gradient
(a) initial segmentation with gradient contours
(b) extended stripes (c) resulting segmentation

5.5 Interpretation of contours

What does a contour map, such as the one in Figure 5.2 (c), tell us about object shapes? Obtained by applying a gradient operation on an intensity image followed by contour tracing and noise removal, image contours occur at places where intensity changes abruptly, such as boundaries of object surfaces, shadows, and marks. There are five types of image contours caused by the intrinsic structure of the 3-D view and occlusion of one object by another. As suggested in [Charniack and McDermott 85], and illustrated in Figure 5.6, these types are:

- extremum* self-occlusion by a curved object, surface gradually turning away from the viewer, surface orientation perpendicular to the viewing direction at the contour.
- blade* two surfaces meet, just one visible; surface orientation discontinuous.
- fold* two surfaces meet, both visible; surface orientation discontinuous.
- shadow* an abrupt change in illumination
- mark* an abrupt change in reflectance or texture due to surface characteristics.

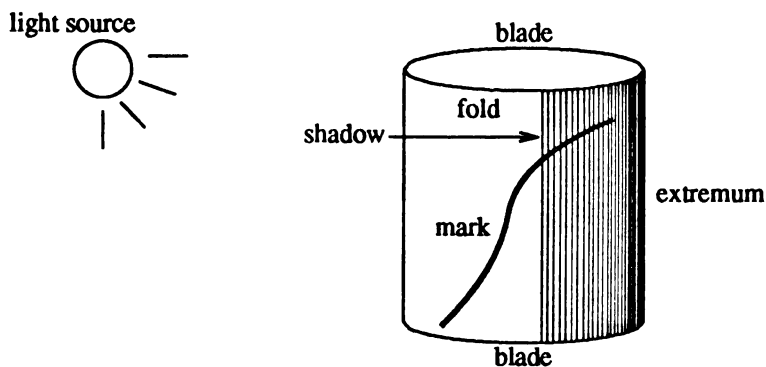


Figure 5.6 Types of image contours

Our goal is to classify the image contours and to infer the surface shapes in the vicinity of the contours.

5.5.1 Apparent curvature of visual contour and local surface geometry

Koenderink [Koenderink 84] proposed a theory that relates the apparent curvature of the occluding contour of a visual shape to the intrinsic curvature of the surface and the radial curvature. According to his theory, convexities, concavities, or inflection of contours in the image allow us to draw inferences about local surface geometry with certainty, assuming objects are *smoothly* curved (e.g. no folds). Two curvatures were defined as follows.

$$\text{radial curvature} \quad : K_r = \frac{1}{r_r}$$

$$\text{transverse curvature} \quad : K_t = \frac{1}{r_t}$$

where r_r is the radius of the curve in which the normal plane cuts the surface, and r_t is the radius of the curve along which the imaging rays cut the projection plane. The normal plane is defined by the normal to the surface at a rim point and the imaging direction, and the projection plane is perpendicular to the normal plane at the rim point. See Figure 5.7. The rim is the boundary between the visible and invisible parts of the object. Note that the rim is not necessarily planar for a general smooth object.

A remarkable simple result is

$$K_r K_t = K$$

where K denotes the Gaussian curvature of the surface, although K_r and K_t are not necessarily the two principal curvatures. If the projection is from a finite distance d , the *apparent curvature* of the contour is

$$K_{app} = dK_t = \frac{dK}{K_r}$$

The explanation of the scaling factor can be found in [Koenderink 84].

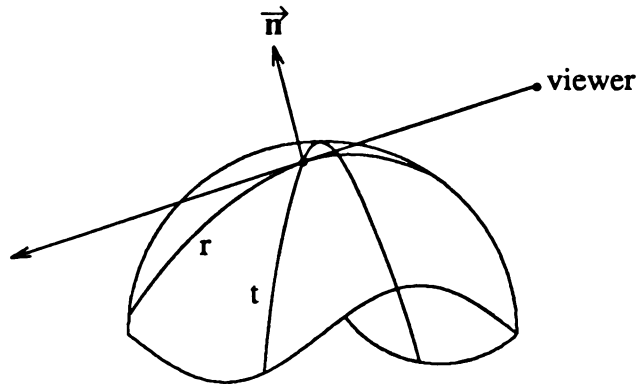


Figure 5.7 Radial curvature and transverse curvature

Notice that the radial curvature K_r cannot change sign at an occluding contour because the object occludes the sight of farther parts. Let K_r be positive (an arbitrary choice), then the apparent curvature K_{app} and the Gaussian curvature K have the same sign. From K_{app} , which can be obtained from the visual contours in the image, we can determine the sign of the Gaussian curvature, i.e. we can draw inferences about the local surface shape. The general rule is : convexities of the contour correspond to convex surface patches, concavities to saddle-shaped surface patches, inflection of contour to inflection ($K = 0$) of the surface. There are no exceptions to this rule.

This theory allows us to interpret a smooth curved object at the rim through the occluding contours available in the image. For example, the contour map of the image of a sculpture and an orange is shown in Figure 5.8, where the labels convex (+), concave (-), or inflection (0) can be assigned along the contours. This is a richer labeling than the

Huffman-Clowes scheme [Huffman 71, Clowes 71], in which occluding contours can only be labeled "occluding" (\rightarrow or \leftarrow).

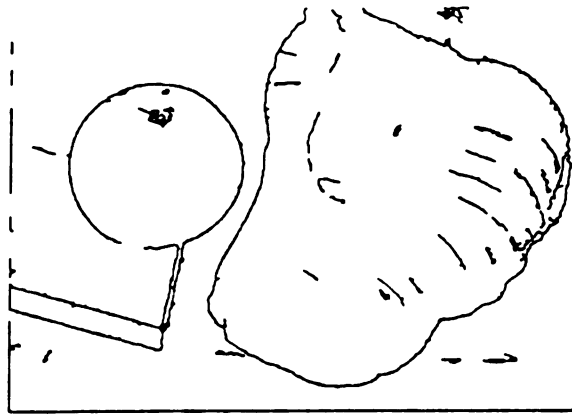


Figure 5.8 Intensity gradient image of a sculpture, an orange and a block

5.5.2 Combination of image contours and stripes

The apparent curvature theory outlined above is valid under the *smoothness* assumption. That is, for rigid objects the tangent plane is defined for every point of the surface and moreover is a smooth function of position. In other words, the theory works on only one type of image contour — the *extremum*. There are contours of other types, such as *blades*, *folds*, *shadows*, and *marks*. For these contours, the apparent curvature theory does not apply. For example, in Figure 5.8, if we know the circular contour at the left of the image is the boundary of a smooth object, as it really is, we can label it convex according to the apparent curvature theory. But if the circular contour were a boundary of a disk, the surface at this boundary would not be convex; instead, it would be a *blade*. To

distinguish *extrema* from *blades*, we may use the *shape from shading* idea [Horn 71] which states that the intensity received is a decreasing function of the angle between the surface normal and the incident direction of the light source. If surfaces are Lambertian, normals can be computed at every point of the surface using intensity information. But the shape from shading technique requires very carefully controlled illumination conditions and uniform surface characteristics (Lambertian, say). It may not be easily guaranteed in many cases. An easier alternative way to decide the type of an image contour is via light stripes.

5.5.3 Terminology and Definitions

First we define some terminology in order to describe more clearly the idea of fusion of intensity and light striping.

The volume viewed by the camera is called the *C-cone* and the volume viewed by the projector is called the *P-cone*. We assume that the objects to be recognized are in both the C-cone and the P-cone so that they are illuminated by the light stripes and visible to the camera. For an object in the C-cone, only the surfaces that are "toward" and closest to the camera are visible. In other words, the surface normal at any point p on the visible surfaces has a positive projection on the imaging vector at p and there is no other surface point on the imaging ray between p and the camera focal point (no occlusion). The visible surfaces divide the C-cone into two parts: the *C-visible part* corresponding to the volume between the surfaces and the camera, and the *C-shadow part* that is the volume at the other side of the surfaces. The boundaries of the object surfaces that are visible to the camera is called the *C-rim*; that is the loci of surface points where the imaging rays are "tangent" (or touch) to the objects. The *C-silhouette* is the image of the C-rim in the image plane. For the projector, the *P-visible part*, *P-shadow part*, *P-rim*, *P-silhouette* are defined similarly.

Under the assumption that the projector lighting is pervasive†, a striped image consists of two kinds of regions: striped regions (stripe networks) and non-striped regions.

The non-striped regions correspond to the P-shadow part in 3-D. We call the non-striped regions in the striped images *P-shadows*. The boundaries of P-shadows, or *P-edges*, according to [L. N. Hambrick, M. H. Loew and R. L. Corroll Jr., 1987], can be categorized into four types as follows:

- (1) shadow-making segment along which the projector light is blocked, labeled *m* segment, which is part of the P-silhouette.
- (2) shadow segment that projects from the *m* segment, labeled *s*.
- (3) occluding segment along which part of the surface is blocked from the camera, labeled *o* segment, which is part of the C-silhouette.
- (4) shadow segment that projects from a *hidden* shadow-making segment, labeled *h*.

These segments of P-shadows are illustrated in Figure 5.9.

For a gray-scale image of the same scene illuminated by a light source at the position of the camera, no C-shadows are perceived. The intensity edges (also called the C-edges) can be one of the following as stated in Section 5.5 :

- (1) occluding edge (extremum or blade, part of the C-silhouette),
- (2) surface normal discontinuity edges (fold),
- (3) surface marks.

† Assume objects in the C-cone are place on a ground plane, so that if there were no objects on the ground plane the image would be filled with a regular grid pattern formed on the the plane.

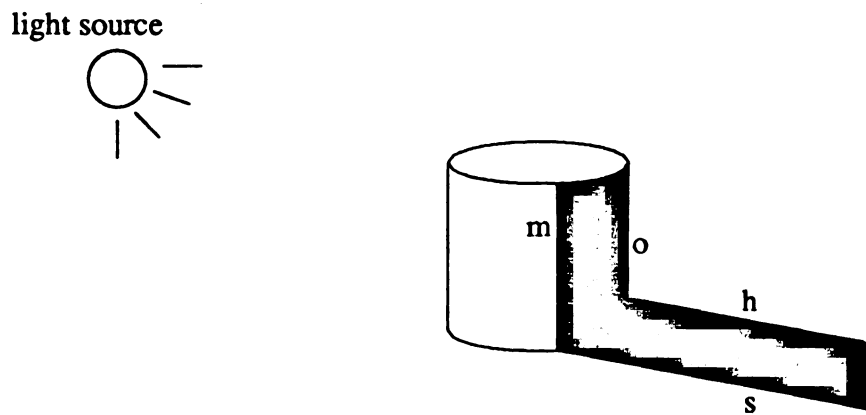


Figure 5.9 Four types of P-edge:
m : shadow-making segment
s : shadow segment
h : shadow segment projects from hidden shadow-making segment
o : occluding segment

5.5.4 Inference Rules

As stated in preceding chapters, stripe networks and stripe texture provide 3-D solutions for grid points, surface normals at these grid points, and rough surface shapes as well. Based on the computed rough shapes, we establish the following inference rules in interpreting adjacent contours under the *general position* assumption. The contours include both intensity edges (C-edges) and virtual edges of the stripe networks (P-edges).

It is clear that some contours in the image do not correspond to geometric characteristics of the objects, such as surface marks of C-edge (due to surface reflectance), s and h segments of P-edge (cast from other segments). These contours, especially the surface marks, are somewhat irrelevant to the object shape. We first establish an inference rule for identifying surface marks.

Rule 1. If a C-edge is crossed by a stripe network, i.e. the stripes crossing the edge are continuous at the edge, and the changes of surface normal at the edge are small, then the C-edge is a *surface mark* (Figure 5.10 (1)).

For s segments (P-shadow segment), they always appear paired with an m segments (P-shadow making segments), as we discussed about the virtual edges in Section 4.3. A virtual edge v of a stripe network is an s segment if there is another virtual edge u of some stripe network such that the stripe end-points on u lie on the projector rays of the stripe end-points of v . That is, the projector rays "enter" the P-shadow at points on u and "exit" the P-shadow at points on v . The two virtual edges u and v is an m - s segment pair that can be determined by the projector geometry[†]. The m - s segment identification is not considered here via inference rules because it is a direct result of 3-D computation rather than 2-D inference, but should be identified before proceeding to apply other inference rules so that the s segments are properly labeled "P-shadow" and will be excluded from further consideration.

[†] It is possible, however, that u is in the C-shadow and hence undetected via 3-D computation.

In what follows in this section (Inference Rules), we assume that the surface marks and the projector-shadows (s segments) are already identified and will be excluded from the rest of the inference procedure (but not to object recognition). The important edge segments are those associated with occlusion, object boundaries, and surface normal discontinuities. The edge segments we will consider include both intensity edges (contours) and virtual edges. The virtual edges are boundaries of the stripe networks and hence part of the P-silhouette, while the intensity edges correspond to the C-silhouette (extremum and blade) and surface normal discontinuity (fold).

The following rules use 2-D stripe networks and rough shape information to make inferences about the types (labels) of the edge segments.

Rule 2. If two striped regions corresponding to two surface patches meet (touch) at a "common" virtual edge and the stripes are staggered from one region to the other, then one surface patch occludes the other from the camera. The virtual edge represents a *blade* if the occluding surface patch is (a) planar or (b) concave, or an *extremum* if it is (c) convex. Usually the virtual edge is also coincident with an intensity edge because it is an occluding edge that is part of the C-silhouette. The occlusion relationship between the two surface patches may be determined by their computed locations in 3-space, or by some global contour and stripe analysis (Figure 5.10 (2)).

Rule 3. If an edge is covered by a stripe network and the stripes at the edge have an inflection (bending stripes), then the edge is a *fold*, because the inflection of the stripes shows discontinuity of surface normals at the edge and both surface patches on the two sides of the edge are visible to the camera. The fold may be an intensity edge and/or a virtual edge. Whether it is a convex or concave fold can be determined by the directions of the surface normals (Figure 5.10 (3)).

The above three inference rules are used for edges "internal" to striped regions (both sides of the edge are striped regions). The next several rules deal with contours at the border of a stripe network, i.e. only one side of the contour is striped.

First, we need to identify contours that are *o* (occlusion) segments. These contours are part of the C-silhouette that occludes objects or part of objects from the camera. There are two types of this kind of occluding contour: one at which the striped surface patch occludes the non-striped one, and the other at which the striped surface patch is occluded by the non-striped surface patch. A contour of the first type is part of the geometry of the striped surface, whereas a contour of the second type is irrelevant to the geometry of the striped surface and needs to be identified early in order not to confuse the surface-edge interpretation.

It is clear that without other information there is no way to tell which side of the contour occludes the other side. Fortunately, the stripe network and the intensity edge provide some clue.

Rule 4. For a one-side-striped edge segment (intensity edge) *e*, if there is another edge segment *r* piercing into *e* to form a "T" junction, then the *r*-side of *e* (T-stem) is occluded by the other side. *r* may be (a) an intensity edge and/or (b) a virtual edge segment (Figure 5.10 (4)).

Rule 5. For a one-side-striped intensity edge segment, if it is part of a relatively closed intensity contour, then the surface enclosed by the contour occludes the other side of the contour (Figure 5.10 (5)).

Rule 4 and 5 are based on some psychological experiences and are similar to some rules used in line-drawing interpretation [Huffman 71, Clowes 71, Waltz 75, Binford and Lowe 85]. In the light striping case, if the non-striped side occludes the striped side, the non-striped side is also associated with an object some of whose surfaces are striped.

At an occlusion edge, if the non-striped side occludes the striped side, this edge is irrelevant to the surface geometry of the striped surface. Hence for surface-edge interpretation, we only consider the edge at which the striped side occludes the non-striped side and edges that are not occlusion segment. In these cases, the striped region in the vicinity of the edge segment is "closest" to the camera — there is no obstacle blocking it from the

camera.

There are two possibilities for such a striped region: the edge segment that bounds the striped region is only a virtual line, or it is also an intensity contour. If it is an intensity contour, it is either part of the C-silhouette or represents surface normal discontinuity, since surface marks are already excluded. If the edge segment is only a virtual line, it is part of the P-silhouette but unlikely part of the C-silhouette. The following inference rules deal with edge segments that are one-side-striped and are "closest" to the camera. The rules can be divided into two groups depending on whether the edge segment is only a virtual line or it is also an intensity contour. See Figure 5.11 for examples where the rules are triggered.

Group 1. Rules in this group apply to cases where the edge segment that bounds a striped region is an intensity contour. A contour is said to be convex if every chord drawn on it lies within the striped region.

Rule 6. A contour c bordering a planar surface (indicated by straight stripes) represents (a) a *fold* if there is evidence indicating an unstriped surface meeting at the contour c , such as another intensity contour at the non-striped side connecting one end of c forming an "L" or an "arrow" junction, or (b) a *blade* otherwise. This is because on a planar surface, the normal to the surface is a constant. At the contour, the normal has to be discontinuous from the visible striped plane to either the invisible surface of the object (blade case) or to the visible unstriped surface of the object (fold case).

This rule can also be stated in terms of the position of the contour relative to the striped region. If the contour bounds the "upper" boundary of the planar surface, it is a *blade* (see Section 5.5.5 for mathematical proof) and hence an o segment; whereas if it bounds the "lower" boundary of the surface, it is either a *fold* or a *blade* — an m segment (P-shadow making segment). The direction indicators "upper" and "lower" were defined in Chapter 3 and are redescribed in Section 5.5.5.

Rule 7. A convex surface patch bounded by a straight contour c is *convex cylindrical* at c . If c bounds the upper border of the surface, which is usually the case since an edge segment at the lower border of a convex surface patch is often a virtual edge but not an intensity contour, c is an o segment and labeled *extremum*.

Rule 8. If a convex surface patch bounded by a convex contour c , the contour is (a) an *extremum* if the surface normal gradually turns away from the line of sight in the direction towards the contour, or (b) a *blade* if the normal near the contour is not perpendicular to the line of sight. In both cases, c is an occluding segment.

Rule 9. A contour c at the lower border of a convex surface patch is either (a) labeled *fold* if an unstriped visible surface meets the contour, or (b) it represents an *extremum* and the surface is *saddle-shaped*. In the former case, c is a P-shadow making segment while in the later case c is an occluding segment.

Rule 10. A contour bounding a concave surface patch is a *blade* or a *fold*.

Group 2. Rules in this group apply to edges that are virtual boundary segments of one-side-striped regions.

Rule 11. Same inference stated in Rule 6 holds for virtual edges that bound planar surface patches. Furthermore, a virtual edge at the lower border of a planar surface is more likely to be a *fold* than a *blade* since a blade is also an occluding segment and occluding segments are often intensity edges.

Rule 12. The virtual edge v at the upper border of a convex surface is an occluding segment and labeled *extremum*; the surface patch at v is either convex if v is convex, or saddle-shaped if v is concave, similar to rule 8 and rule 9.

Rule 13. The virtual edge v at the lower border of a convex surface patch is a P-shadow making segment.


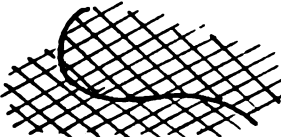



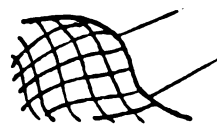

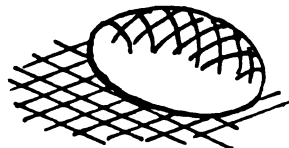
example figure	stripe network	contour segment	surface type at contour	contour label
1 	stripe continuous, normal same	all kinds	any kind	surface mark
2a 	planar occluding	all kinds	planar	blade
2b 	concave occluding	all kinds	concave	blade
2c 	convex occluding	convex	convex	extremum
3 	stripe continuous, normal change	all kinds	corner	fold
4a 	all kinds	intensity edge T junction	any kind	occluding
4b 	all kinds	virtual edge T junction	any kind	occluded
5 	all kinds	part of closed contour	any kind	occluded

Figure 5.10 Inference rules 1 — 5

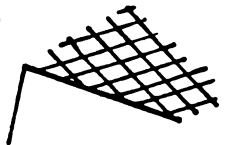
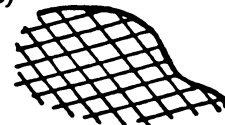
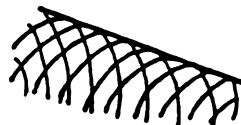


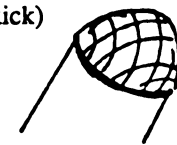



example figure	stripe network	contour segment	surface type at contour	contour label
6a (step) 	planar	all kinds	planar	fold
6b (knife) 	planar	all kinds	planar	blade
7 (can) 	convex	straight	developable	extremum
8a (ball) 	convex, large change of normal	convex	convex	extremum
8b (can) 	convex, small change of normal	convex	convex	blade
9a (lip stick) 	convex, no curvature towards contour	convex	convex	fold
9b (donut) 	convex, curvature towards contour	concave	saddle	extremum
10 (bowl) 	concave	all kinds	concave	blade or fold
13 (ball) 	convex	all kinds	lower border	P-shadow making segment

Figure 5.11. Inference rules 6 — 13

5.5.5. Some mathematic arguments about the inference rules

In Chapter 3, we defined direction indicators "up" and "down", or "upper" and "below", in the image plane in terms of the spatial relationship between the camera and the projector. The projecting rays in the 3-space emitting from the projector are defined with direction. Accordingly, the projection of these rays in the image plane also defines the direction in the image. The direction in the image is from "up" to "down", or from "upper" to "lower". If a striped region is bounded by a contour segment, then the contour segment is said to be the "upper" boundary if at every contour point the *direction* is from the non-striped side to the striped side of the contour; and said to be the "lower" boundary if the direction at every contour point is from the striped side to the non-striped side.

Assumption A. In the following two lemmas, we only consider surface patches that are one-side-striped, "closest" to the camera (not occluded by the non-striped region at the other side of the edge), and their attached edges are neither P-shadow edges nor surface marks.

Lemma 5.1 Under the *Assumption A*, the upper boundary of a planar surface patch represents a *blade* as stated in rule 6.

proof. Since the non-striped side of the edge has no stripes, there are two possibilities: (1) either the side is invisible to the camera and hence the stripes at that side cannot be seen; in this case, the edge is obviously a blade; or (2) the side is invisible to the projector but visible to the camera. In this case, one of the following two situations must occur : (a) it is occluded by an other object with respect to the projector (P-shadow) or to the camera, but this kind of occlusion has already been identified in an earlier stage under *Assumption A* and excluded here; or (b) it is occluded by the object itself. Here "occlusion" is with respect to the projector. This is illustrated in Figure 5.12.

Let the "projector vector" (reverse direction of the projecting ray) at the edge point A be \vec{p} , the normal of the non-striped surface (other side of the edge) at A be \vec{n} , and the imaging vector be \vec{c} . Let P be the plane perpendicular to \vec{p} and N be the plane

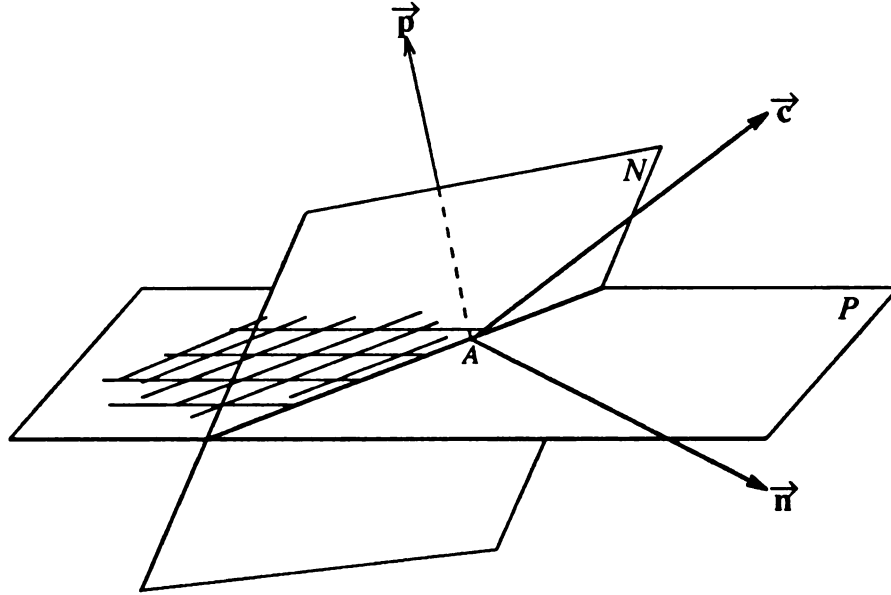


Figure 5.12 Lemma 5.1

perpendicular to \vec{n} . Since the non-striped side of the edge is occluded by the object itself, \vec{n} and \vec{p} must be at the different sides of the plane P and plane N , i.e. their angle is greater than 90° . But, since the camera sees both sides of the edge, \vec{c} must be on the same side of P as \vec{p} , and at the same side of N as \vec{n} . That is, \vec{c} is in the "quadruple" between plane P and plane N . Hence, the projection of \vec{p} in the image plane (from \vec{p} toward \vec{c} in Figure 5.12) would define the edge the "lower" boundary of the striped region, which contradicts the condition that the edge is the "upper" boundary of the region. So, the non-striped side of the edge cannot be occluded by the object itself with respect to the projector. This excludes the possibility (2) and the lemma follows. ■

Lemma 5.2 Under *Assumption A*, if a striped region bounded *below* by an edge indicates a smooth curved surface patch, the non-striped side of the edge is visible to the

camera.

proof. Refer to Figure 5.13, let the projector vector and the imaging vector be \vec{p} and \vec{c} , respectively, and the surface normal at an edge point A be \vec{n} . Since the surface is smooth and the stripes stop at the edge, $\vec{p} \cdot \vec{n} = 0$ (they are perpendicular). Because point A is visible to the camera, $\vec{c} \cdot \vec{n} \geq 0$. We now prove the equality does not hold, hence $\vec{c} \cdot \vec{n} > 0$. Suppose $\vec{c} \cdot \vec{n} = 0$. Since both \vec{p} and \vec{c} are perpendicular to the surface normal \vec{n} at A , they span the tangent plane of the surface at A . The object in the vicinity of A is at one side of the tangent plane. Since the projection of \vec{p} in the image plane, \vec{p}_i , is the intersection of the image plane and the \vec{p} - \vec{c} plane, the striped region at the vicinity of A would also be at one side of \vec{p}_i in the image. That is, \vec{p}_i is "tangent" to the striped region at A_i which is the image of the edge point A . But \vec{p}_i should run across the edge at A_i because the edge is the "lower" boundary of the striped region. So $\vec{c} \cdot \vec{n} > 0$ follows the contradiction. Hence, not only the edge point A is visible to the camera, so is a vicinity of A , including the non-striped side of the edge at A . ■

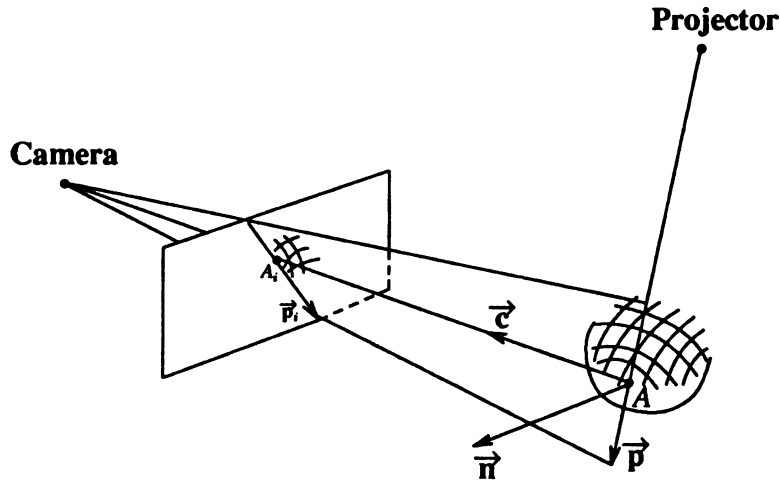


Figure 3.13. Lemma 5.13

From Lemma 5.2, we know that if a virtual edge is the lower border of a smooth curved surface patch (indicated by a striped region) the non-striped side of the edge is also visible to the camera. This leads us to search for intensity edges down the area that may provide "real" surface boundary information about the non-striped surface patch.

For other inference rules, some are heuristic, and some others can be proved mathematically.

5.5.6 Implementation of the inference rules

The implementation of the inference rules may be done via a set of production rules and a control scheme. These production rules may be used to form a decision tree that controls the labeling process. The input and output are the following :

input :

- 1). contours in some form (chain-code, say),
- 2). surface patches, their types and boundaries;

output :

- 1). labels associated with each contour (label can change along a single contour),
- 2). surface types.

The labeling process is a contour-based procedure. It takes a contour at a time, segments the contour based on its adjacency relationships with the striped regions, and labels each segment of the contour according to the inference rules. Since the production rules are triggered in a specified order, surface marks and *m-s* segments are determined first. Occlusion with respect to the camera is then determined and only these striped surfaces and corresponding edges that are "closest" to the camera are processed, since otherwise the edges are not very useful in inferring the surface geometry. Then the rules in group two are applied. As output, each contour segment may have one or more labels associated with it.

Multiple labels can be disambiguated using the knowledge about the spatial relationship between the camera and the projector. For example, if a contour in the image bounds the "upper" part of a planar surface, it indicates a *blade*, rather than ambiguous labels "blade or fold", as claimed in Lemma 5.1. In addition a relaxation procedure based on the relationships between adjacent image contours can also be used to sort out some multiple labels.

5.5.7 Examples

We take two scenes as examples. The first is the image of a solid block as shown in Figure 5.1, in which all surface patches are planar. The second example scene consists of a cobra sculpture, an orange and a block, in which surface patches are curved. The labeling procedure for the surface patches and associated edges in these two examples are done manually — a human account of what coordinated use of individual processes and rules developed in Section 5.5.4 *might* produce.

example 1 — the solid block

The intensity gradient contours and the striped networks are reprinted in Figure 5.14.

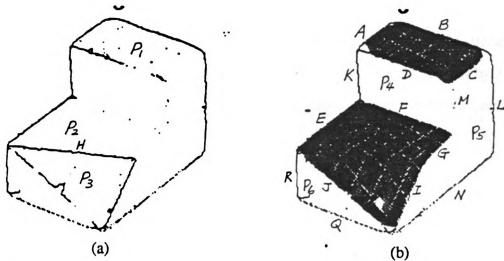


Figure 5.14 Image of a solid block

(a) intensity gradient contours

(b) overlay of contours and stripe networks with named surface patches and edges

There are two striped regions, but the lower one can be divided into two sub-regions by the bending of the stripes and the intensity contour H at the bending place. Let the three striped surface patches be P_1 , P_2 and P_3 , and the contour segments be A , B , C , ..., R , as

indicated in Figure 5.14(b). First, by 3-D relationships between the virtual edges, F is a P-shadow cast by D . By the inference rule 3, H is labeled "fold". Furthermore, using the 3-D information (normals), H is classified as a "convex fold". According to the inference rule 6 and Lemma 5.1, contours A, B, E are labeled *blade*, whereas C, D, G, I, J are labeled "fold or blade". By rule 6, since D, A and a third contour K (not adjacent to a striped region) form an "arrow", D must be a fold, either convex or concave. But it cannot be concave because it generates the shadow F . So, D is a convex fold, similarly so are C and J . For edges D and K , it is very likely that they form a plane P_4 , that can be verified by the intensity in this area. Similarly, plane P_5 formed by straight lines C and L is decided. Since the intensity contour N connects to L and is parallel to C , it is determined to be in the same plane P_5 . Because of the plane P_5 , the contours G, I are determined *fold* by rule 6. Since P_4 and P_5 are planar surfaces and adjacent to P_1 at fold edge C and D , respectively, they are very likely to intersect at a common edge which starts at the junction of C and D and is actually indicated by the weak intensity edge M . Because both P_4 and P_5 are visible to the camera, M is a *fold*. Finally, plane P_6 formed by J, Q and R is claimed. Now, for the block, we obtained the following representation (including processing described in Chapter 3):

6 planar surfaces:

P_1 — blades A, B , convex folds C, D

$z=80\text{mm}$ in the global coordinate system within the x-y domain defined by stripe end-points (specific numbers are omitted here)

P_2 — blade E , P-shadow F , convex folds G, H

$z=38\text{mm}$ in the global coordinate system within the x-y domain defined by stripe end-points (specific numbers are omitted here)

P_3 — convex folds H, I, J , adjacent to P_2 at the common edge H ,

surface normal perpendicular to the planar surface that is determined by fitting a plane for the stripe grid points.

P_4 — convex fold D, M , occluding contour K and P-shadow F ; adjacent to P_1 at common edge D to P_5 at M

P_5 — convex folds C, G, I , occluding contours L, N , and fold M ; adjacent to P_1 at C , to P_2 at G , to P_3 at I , and to P_4 at M

surface normal to P_5 determined by the 3-D stripe end points on C, G and I

P_6 — convex fold J , occluding contours Q and R , adjacent to P_3 at J

example 2 — a cobra sculpture, an orange, and a block

The gradient contours and stripes plus contours are shown in Figure 5.15 (a) and (b) respectively.

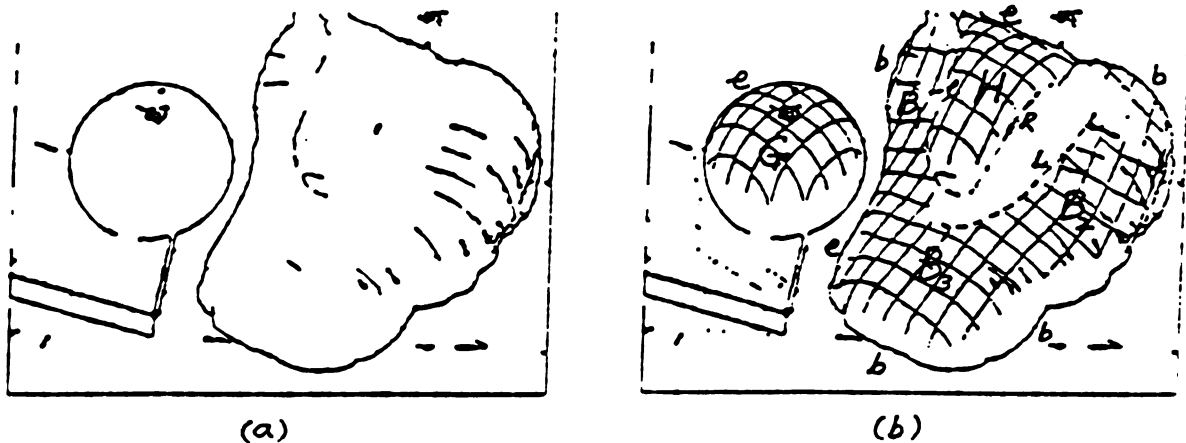


Figure 5.15. Image of a cobra sculpture, an orange and a block

(a) intensity gradient contours

(b) overlay of contours and stripe networks with named surface patches and labeled edges (e —*extremum*, b —*blade*)

There are 3 stripe networks in the image; G for the orange, B for the cobra body, and H for the cobra head. Since there are no stripes on the surfaces of the block, it cannot be interpreted using the fusion rules that require stripe information, so it will not be discussed here. The surface patch G is determined to be spherical by the B-spline fitting of 3-D data, and the boundary of it is *extremum* according to the inference rule 8. Surface patch H is convex by the stripe texture analysis; its upper bordering contour is *extremum*. The left border of H consists of a broken contour which is bridged using the virtual edge of the stripe network; and is determined *extremum* also. For the surface B , both surface fitting and texture analysis suggested that it should be interpreted locally. Roughly, B can be subdivided into three patches B_1 , B_2 and B_3 as indicated in the Figure 5.15, where B_1 and B_2 are basically concave whereas B_3 is convex. Part of the boundaries of B are labeled as indicated in the figure. Notice that the lower borders (virtual edges) of B_2 and B_3 are "extremum" with respect to the projector and hence P-shadow making segments according to rule 13. Also they have small z-coordinate values (5 - 15mm). The intensity edges further down are found that indicate the real surface boundaries of the surfaces, and very close to the ground plane ($z=0$), so B_2 and B_3 are labeled *blade*. Those intensity contours interior to the striped regions are all *marks* by the inference rule 1. For the virtual edges R (at the right border of H) and L (at the left border of B_2), it is determined that L is the P-shadow cast by R . Also, the surface patch H occludes B_1 according the inference rule 2.

The representation of this image can be described as follows.

There are five surface patches:

- G — convex sphere with radius 36mm (by B-spline and Gaussian curvature computation), at 3-D location determined by the stripe grid points (specific numbers were not computed), bordered by extremum contour;
- H — convex, occluding surface patch B_1 at left extremum contour;

right virtual edge casts shadow at the left of surface patch B_2 ;

B_1 — basically concave, occluded by H , boundaries labeled as indicated in the figure;

B_2 — basically concave, boundaries labeled as indicated in the figure;

B_3 — convex, boundaries labeled as indicated in the figure,

B_1 , B_2 and B_3 being on the same surface, with locations in the global coordinate system determined by their grid points in the 3-space.

5.5.8 Experiments

Five more images of multiple object scenes were analyzed and the surface patches and edges were labeled using the inference rules. The inference rules were coded in PROLOG. The input to the PROLOG predicates was a set of hand-coded "facts" about the edges and surface patches in terms of their geometric and topological properties (adjacency, planar-ness of surface patches, etc.). The scenes were rather complex. Although the representations were hand-coded, we assume that they may be derived from the lower level processing described in Chapter 3 and 4. The PROLOG coding of the inference rules and the output labels of the five images are listed in the Appendix. The results are summarized in Table 5.2. The results show that 75% of the edges (101 out of 135) were labeled and only one mislabeled. That mislabeled edge is part of a "fold", but due to lack of fold evidence it was labeled "blade". This mistake can be corrected by a higher level process that uses some connection rules and segments these edges and surface patches into objects. The remaining 25% of the edges were not labeled; for two of them, no conclusion can be drawn from the inference rules because there is not enough information about the occlusion relationship between the two sides at the edges. The rest of the edges are simply not associated with striped surface patches. The occlusion relationships between the 26 surface patches were also correctly discovered.

Table 5.2 Summary of results of labeling surface patches and edges of five scenes using the inference rules

objects in the scene	input		output			
	number of surfaces	number of edges	number of surfaces labeled	occluded† surfaces discovered	number* of edges labeled	number of edges mislabeled
book orange metronome	5	28	5	2	24	0
shoe bowl box	6	22	6	2	22	0
can cup cup with cover	5	32	5	2	16	0
tomato can vacuum cleaner	5	26	5	3	21	1
tea pot plate roll of paper	5	27	5	3	18	0

† All occlusions discovered are correct.

* Only edges that are associated with striped surfaces are labeled.

5.6. Conclusions

This chapter investigated a multi-sensor fusion approach that combines information from both the striped image and the intensity image. The two images can easily be accurately registered and hence can be treated as a single image. The gray-tone image is used primarily for extracting gradient contours, while the stripe image provides 3-D surface shapes. Object contours are very much essential to visual perception. Humans often can recognize objects using only contours. For smooth object surfaces, some occluding contours can be interpreted "convex", "concave" etc. according to Koenderink's apparent curvature theory. But, for objects that are not smooth, or contours that are not occluding ones, we need information about the surface itself. Light striping is just the very means that meets our needs. Hence, fusion of intensity contour and striped surface is a natural and easy way that can be used to obtain a good scene representation and low level interpretation.

Light striping offers shape information mostly interior to the visible surfaces; the information about the boundary areas is often quite vague, because of either fading stripes at extrema and hence less accurate computation toward the boundaries or no boundary information available from surface fitting. On the other hand, intensity contours provide just the complement. To combine them in constructing a reasonable object representation that is appropriate for model matching, we developed a set of inference rules that label the surface patches and contours according to their detected features and their spatial relationships. Although some are heuristic and empirical, most of these rules are mathematically elegant and performed well in experiments.

Almost all of the contour labels assigned by the rules are useful for object model indexing, and some are useful in matching to recognize and determine pose. In the future, we hope to add more rules to help in further disambiguation and segmentation of the scene into objects. It is important to note that our approach has been heuristic — we chose not to rigidly define the class of objects and we have relied on general viewpoint

assumptions, which will not always hold true.

CHAPTER 6

Concluding Discussion and Future Work

This thesis generalized use of striped light toward obtaining a $2\frac{1}{2}$ -D representation of a scene to be used ultimately for object recognition and pose detection. This was demonstrated by using structured lighting in three stages in 3-D vision, namely 3-D sensing, surface shape inference through stripe textures, and $2\frac{1}{2}$ -D representation via fusion of striped images and gray scale images. This chapter summarizes the work described in this thesis, discusses its implications, and suggests directions for future work.

6.1 Summary

The first chapter laid the groundwork for 3-D machine vision by briefly discussing the three basic parts of 3-D vision: feature extraction, object representation, and model matching. 3-D data acquisition, which provides spatial information about the object surfaces, is one of the first tasks in feature extraction. From 3-D data, geometrical relationships between surface points can be established, which are used for describing surface shapes. In object representation, surface shapes must be inferred and represented in an appropriate format in order that the object be recognized by matching its representation against some predefined models. Many shape-from-x techniques have been developed in the past two decades that infer surface shapes using information from different sources. Structured light is one of these sources, and it is the one investigated in this thesis.

Model matching is an object recognition procedure that matches the representation of the sensed data to a stored representation of the object. Constructing models automatically is itself a very difficult problem; and to develop a very large model base is even harder. This thesis concentrates on the first two tasks in 3-D vision, namely feature extraction and construction of surface representation. In particular, a structured lighting approach is developed.

The structured lighting approach in a general context was discussed in Chapter 2. Structured lighting is a technique that imposes artificial features on the surfaces of the objects by illuminating the scene with a *structured* light pattern. When using a grid as the illumination pattern, the light stripes weave "networks" on the surfaces that give an image where the grid distortion is a function of the geometric shapes of the surfaces, as well as the spatial relationships between the sensor, the light source, and the object surfaces. The light source (a projector) and the sensor (a camera) form a stereo system in which surface points in 3-space can be solved through triangulation. The major difficulty is the stripe identification problem which is the general "correspondence problem" in the structured light setting. This problem is attacked by applying a set of general constraints to the computed surface points that relate to each other in the stripe networks. In addition to 3-D data, light striping also provides rich information about surface shapes through the 2-D stripe textures. Analysis of stripe texture may confirm the surface shapes obtained from the 3-D data, and hence increase the certainty of inference about surfaces. It is apparent that the sensed 3-D points are sparse due to the way the grid of light is projected. We must assume the "smoothness" of the surfaces in order to make inferences about the surfaces using only the sensed data, which may not be valid in general but is often satisfied in practice such as in industrial environments. To make surface inferences more reliable, and to obtain information in the non-striped regions, structured lighting is fused with the gray tone image of the same scene. Surface-based representation may be constructed based on the combination of the two information channels.

6.1.1 Computing 3-D surface solutions

Chapter 3 is dedicated to 3-D sensing using light striping. The striped image is first thresholded to generate a binary image, which is then processed to extract the grid points (stripe intersection and end points). For those grid points extracted, a stereo computation is carried out for their locations in the 3-D space. The stereo pair here is the image sensed by the camera and the grid slide that is projected onto the scene. In order to use stereo, the spatial relationships between the projector, the camera and the scene must be fixed and known in advance: this is accomplished through a process commonly referred to as the calibration process. After the projector and the camera have been calibrated, a point in the 3-D space can be computed via triangulation if it is identified in the stereo pair. Identifying matching points, or solving the stripe labeling problem, is very difficult when the stripes are generated from a homogeneous pattern (a grid, for example). But unlike the general "correspondence problem", not only does light striping substantially reduce the search space but also it makes the relationships between the stereo matching points more explicit and easy to resolve. Two general constraints — the uniqueness constraint and the continuity constraint are established and are further refined into a set of geometric rules and a set of topological rules that are specially derived for solving the line labeling problem for the stripe networks. The relationships of the grid points within and between the stripe networks are specified by these rules that are obeyed when the striped images are formed under the general position assumption. A graph traversal procedure is carried out to propagate the constraints within a stripe network to sort out those stripe labels that violate any of the constraints. A global constraint test is then applied to relate several individual stripe networks in a multi-network image, that further reduces the ambiguity in stripe labels. Five algorithms are described in detail which applied in sequence will take the sensed striped image as the input and produce 3-D surface solutions for the grid points in the stripe networks. The five algorithms are: 2-D processing for extracting grid points of interest, 3-D computation for a single grid point via

triangulation, within-network constraint propagation, stripe network boundary extraction, and between-network constraint propagation. Time complexity was analyzed. Experiments were performed on various real images. The results of three of these images are listed in Table 3.2. These images contain both polyhedral and curved objects of arbitrary poses and occlusion relationships, some of which have complex shapes (e.g. cobra sculpture). The experiments showed that the resulting 3-D surface solutions delivered by the five algorithms are of reasonable accuracy, and the small degree of ambiguity which remains may further be reduced using other information. Each of the 3-D surface solutions can be used in inferring surface shapes. A mathematical analysis showed that the degree of ambiguity in the stripe labeling is proportional to the camera standoff (distance from the camera to the scene point) and the error tolerance in computation, and inversely proportional to the stripe spacing.

The conclusion from this work on light striping in 3-D sensing is that it is effective, reliable, and easy to apply, but yields a small degree of ambiguity in some surface locations.

6.1.2 Classification of surface patterns

3-D surface data by themselves do not compose the surface description for scene understanding; they are still a low-level representation of the objects but can be used in deriving a higher level surface description. In addition to 3-D surface solutions, structured lighting also directly reveals surface shapes through the stripe networks imposed on the surfaces. These stripe networks make the surfaces vividly visualized to a human due to the way the stripes (surface contours) are formed. Chapter 4 discussed surface classification using 3-D data and stripe textures.

Based on the computed 3-D points, a surface can be fitted using the B-spline technique, which was the first topic in Chapter 4. Instead of generating an analytical form of the fitted surface, the two principal curvatures and the intrinsic Gaussian curvature were

calculated at various points interior to the stripe network. The curvatures were then clustered and the surface patch (corresponding to a stripe network) was classified according to the curvature clusters and classical differential geometry theory. Several surface patches obtained from the real striped images were measured in terms of the curvature properties. The experiment was described in Section 4.1.3. These surface patches (a planar surface, a Coke can, and a potato) were correctly labeled *planar*, *cylindrical*, and *elliptic*. Furthermore, the radius of the cylindrical surface (the Coke can) was also successfully estimated using the principle curvature measures.

A second direction in surface shape classification is through the 2-D stripe networks, or 2-D stripe textures. Since the stripe texture in the 2-D image is a distorted version of the grid on the projector slide, and the way the grid is distorted is a function of the surface shapes, we may recover the surface shapes, to some extent, by just looking at the 2-D stripe textures. Planar surfaces can be identified by the straight stripes in the image; convex and concave surfaces can be determined by the way the stripes are curving in the 2-D image. Texture changes in certain directions (X-stripe direction, Y-stripe direction, and diagonal directions) are also useful in determining the surface variance along these directions, which may be used in the fusion of striping and intensity. In an experiment stated in Section 4.2.4, 25 of 27 object surfaces were correctly classified as *planar*, *convex*, *concave*, and *irregular*, using only 2-D information from the striped networks. Also seven spherical or cylindrical surfaces were distinguished using texture gradient measures. It is convincing that the 2-D stripe patterns make available a rich piece of information for surface interpretation that may work alone without support of 3-D data.

6.1.3 Fusion of intensity and stripe image data

A fusion scheme that combines information obtained from light striping and intensity was introduced in Chapter 5. As is often the case, the intensity image provides edge information that is important for edge-based segmentation, which some psychologists

claim [Biederman 87] is the primary information source for human perception. As the dual of the striped image, an intensity image of exactly the same scene can be easily obtained by simply switching off the projector and turning on a light at the position of the camera. This intensity image offers gradient contours that significantly complement what is in the striped image. But edge detection is itself very noise sensitive and it is usually very hard to extract perfect edges. On the contrary, the striped image offers good surface information and is much more noise insensitive. To overcome the weaknesses of each of the two types of images and facilitate their strengths, fusion of the two was investigated for achieving better segmentation and constructing surface-edge representations. A set of rules were developed that make inferences about striped surface patches and related edge segments and assign labels (extremum, blade, fold, shadow, mark) to the edges obtained from the intensity gradient. In the experiments reported in Section 5.5.8, a set of PROLOG predicates that implements the rules was applied to five images of multiple object scenes. Out of 101 edges only one was mislabeled. The surface labels assigned by the surface analysis in Chapter 4 and the edge labels assigned by the fusion rules in Chapter 5 are useful for object model indexing, and some are useful in matching to determine object pose and recognition.

From the fusion of the light striped image and gray-tone image, we reach more certain and reliable object descriptions. There are some other information sources that may be useful, such as a second striped image taken by projecting a grid from a different angle (camera not moved), that may put stripes on the areas that were P-shadows before.

6.2. Conclusions

This thesis has investigated the usefulness of structured lighting in machine vision for 3-D scene representation and ultimately for object recognition. In a controlled environment such as an industrial robot workspace, a single grid-illuminated image can be used to sense the 3-D surface points in a rather efficient way as long as the objects are

relatively smooth and not too small to be covered by multiple stripes. Since our sensing system is a "snapshot" system rather than a "scanning" system, it has potential speed and cost advantages. The constraints employed in developing the surface solutions are from the real world and are very powerful in solving the stripe identification problem. Although the resulting surface solutions may not be unique in general, the degree of ambiguity is often small in practice and each of the possible locations in the 3-space serves equally well in shape analysis. For object grasping, the ambiguity can be resolved via tactile sensing.

Not only is the structured lighting a tool for 3-D sensing, but it also provides a framework in which surface shape can be inferred and the scene representation can be established. The projected grid is easy to generate. The image processing task is simplified[†] because only one striped image is required and the grid is a uniform pattern. The 2-D stripe networks clearly reveal the surface geometry through their texture variation. A gray scale image of the same scene is obtained by simply turning off the projector, and hence it can be exactly registered with the striped image. The two images provide information where both edge-based techniques and region-based techniques can be applied.

The advantages of using structured light in 3-D scene analysis are balanced by some restrictions in implementation. First, it is not trivial to process the striped image. To extract the bright stripes, we must assume little interference from other factors, such as no strong reflection from the object surface, otherwise the strong reflection surface spots might be treated as part of the stripes. Second, accidental alignment of stripes can not always be avoided; it occurs infrequently but not rarely, especially when stripe spacing is small for obtaining denser data. Theoretically, this will not be a fatal problem since when no surface solution can be found (this is often the case when two or more surface patches

[†] but is still not easy

are treated as one single surface patch due to accidental stripe connection) we can always break the network into pieces and apply the algorithms to each piece. But this will considerably complicate the process because deciding how to break the large network into small ones is itself not trivial. The third restriction lies on the ambiguity in the location of 3-D surface solutions. However, this restriction is generally not a severe problem since the degree of ambiguity is usually small and can be further reduced by additional knowledge which is often available in practice. Also, as far as 3-D surface shape is concerned, any of the multiple solutions works.

6.3. Future work

The research presented in this thesis can be carried forward in several ways: the theoretical understanding of the structured lighting technique can be improved, the algorithms themselves can be improved, the fusion scheme can be integrated into an expert system for generating symbolic representation, and the representation should be used for modeling and matching. Results of this thesis provide a method which could provide the basis for development of a practical 3-D sensor system for bin-picking.

The theory of this research lies in the field of geometry (analytical and differential), and classical image processing as well. In 2-D processing, we need a general approach to extract the stripes, particularly for those stripes that are very close to each other (when the line of sight of the camera is almost parallel to the surface). The geometric and topological constraints discussed in Chapter 3 and the inference rules for fusion in Chapter 5 are for ideal mathematical cases. Violations to these rules may occur when they are applied to real images, although they will occur infrequently. Hence, *uncertainty* might be taken into consideration, or feedback from other processes needs to be integrated into this step.

The algorithms were developed with no special attention to efficiency. For example, files were created to hold intermediate results that were read as the input by the next

algorithm in the processing sequence. Sometimes routine work was done by hand rather than by programs. This scheme may be improved using pipeline-like processing to save disk I/O. The algorithm 5 in Chapter 3, which projects striped regions back to the slide plane and finds overlap, is combinatorially complex. This complexity may be reduced by applying some rules such as "overlapping is checked only for adjacent regions" and the like. The B-spline fitting algorithm needs to be supplemented by some other surface fitting method that can handle smaller surface patches or surface borders.

Computer algorithms for the fusion of intensity and stripes have not yet been completely implemented. The algorithm for that may be part of an expert system, but the detailed steps (for example, to segment contours, to relate striped regions to contour segments etc.) need to be implemented before the symbolic processing can be carried out.

Finally, for the ultimate goal of object recognition, a model matching step is needed. How structured lighting techniques can be employed in generating models automatically and how the matching would be performed based on the surface-edge representation is the topic of current research. Research on this topic has been carried on for a few years in the PRIP lab at Michigan State University [S. W. Chen and Stockman 87]. That work has to be investigated with what has been discussed in this thesis to complete the basis for a 3-D machine vision system.

APPENDIX

APPENDIX

1. Inference rules coded in PROLOG

	<code>borders(E,R)</code>	<code>:- lower_borders(E,R).</code>
	<code>borders(E,R)</code>	<code>:- upper_borders(E,R).</code>
	<code>planar_region(R)</code>	<code>:- planar(R);ground_plane(R).</code>
1.	<code>mark(E)</code>	<code>:- interior(E,R),stripes_smoothly_cross(R,E).</code>
	<code>pshadow(E)</code>	<code>:- shadows(E1,E).</code>
13.	<code>shadow_making(E)</code>	<code>:- shadows(E,E1).</code>
	<code>occludes_at(R1,R2,E)</code>	<code>:- ground_plane(R2),two_side_striped(E,R1,R2).</code>
4a.	<code>occludes_at(R1,R2,E1)</code>	<code>:- borders(E1,R1),t_junction(E2,E1),borders(E2,R2).</code>
4a.	<code>occludes_at(R1,R2,E1)</code>	<code>:- t_junction(E2,E1),borders(E2,R2), two_side_striped(E1,R1,R2).</code>
	<code>occluded_at(R,E)</code>	<code>:- occludes_at(R1,R,E).</code>
4b.	<code>occluded_at(R,E)</code>	<code>:- one_side_striped(E,R),borders(E1,R),t_junction(E1,E).</code>
4b.	<code>occluded(R)</code>	<code>:- occluded_at(R,E).</code>
	<code>stage2(E)</code>	<code>:- not (mark(E)),not (pshadow(E)).</code>
	<code>stage3(E)</code>	<code>:- region(R),stage2(E),not (occluded_at(R,E)).</code>
2a.	<code>blade1(E)</code>	<code>:- two_side_striped(E,R1,R2),stagger(R1,R2), occludes_at(R1,R2,E),(planar(R1);concave(R1)).</code>
2b.	<code>blade1(E)</code>	<code>:- two_side_striped(E,R1,R2), stagger(R1,R2),concave(R1),region(R1),region(R2).</code>
6a.	<code>blade1(E)</code>	<code>:- planar(R),lower_borders(E,R), not (arrow(E1,E,E2),not (borders(E1,R),borders(E2,R))).</code>
6b.	<code>blade1(E)</code>	<code>:- planar(R),upper_borders(E,R),one_side_striped(E,R).</code>
8b.	<code>blade1(E)</code>	<code>:- borders(E,R),convex(R),convex(E), normal_constant_to_edge(E,R).</code>

10a.	blade1(E)	:-	upper_borders(E,R),concave(R).
10c.	blade1(E)	:-	concave(R),lower_borders(E,R), not (arrow(E1,E,E2),not (borders(E1,R),borders(E2,R))).
	blade(E)	:-	blade1(E),stage3(E).
3.	fold1(E)	:-	two_side_striped(E,R1,R2),stripes_continuous(E), normal_change_2(R1,R2,E).
3.	fold1(E)	:-	interior(E,R),edge(E),region(R),normal_change_1(R,E).
6a.	fold1(E)	:-	lower_borders(E,R),l_connects(E,E1),not borders(E1,R).
6a.	fold1(E)	:-	planar(R),lower_borders(E,R),arrow(E1,E,E2), not (borders(E1,R),borders(E2,R)).
9a.	fold1(E)	:-	lower_borders(E,R),convex(R),l_connects(E,E1), not (borders(E1,R)).
9a.	fold1(E)	:-	lower_borders(E,R),convex(R),arrow(E1,E,E2), (not borders(E1,R);not borders(E2,R)).
10b.	fold1(E)	:-	concave(R),lower_borders(E,R),arrow(E1,E,E2), not (borders(E1,R),borders(E2,R)).
	fold(E)	:-	fold1(E),stage3(E),not occluded_at(R,E).
2c	extremum1(E)	:-	two_side_striped(E,R1,R2),edge(E),region(R1), region(R2),occludes_at(R1,R2,E),staggers(R1,R2), convex(R1),not normal_constant_to_edge(E,R1).
7.	extremum1(E)	:-	convex(R),straight(E),intensity(E), borders(E,R),region(R),edge(E).
8a.	extremum1(E)	:-	convex(R),borders(E,R),convex(E),intensity(E), normal_changes_to_edge(R,E),region(R),edge(E).
9b.	extremum1(E)	:-	borders(E,R),convex(R),concave(E), intensity(E),region(R),edge(E).
12.	extremum1(E)	:-	upper_borders(E,R),convex(R),virtual(E), region(R),edge(E),not intensity(E).
	extremum(E)	:-	extremum1(E),stage3(E).
9b.	saddle(R)	:-	borders(E,R),convex(R),concave(E),intensity(E).
7.	cylindrical(R)	:-	convex(R),straight(E),intensity(E),borders(E,R).

2. Representation primitives of scene 1

edge(e1).	convex(e1).
edge(e2).	convex(e2).
edge(e3).	convex(e14).
edge(e4).	convex(e16).
edge(e5).	convex(e18).
edge(e6).	convex(e19).
edge(e7).	convex(r2).
edge(e8).	convex(r4).
edge(e9).	concave(r1).
edge(e10).	concave(r3).
edge(e11).	
edge(e12).	straight(e6).
edge(e13).	straight(e9).
edge(e14).	straight(e10).
edge(e15).	
edge(e16).	intensity(e1).
edge(e17).	intensity(e2).
edge(e18).	intensity(e3).
edge(e19).	intensity(e4).
edge(e20).	intensity(e6).
edge(e21).	intensity(e7).
edge(e22).	intensity(e8).
edge(e23).	intensity(e10).
edge(e24).	intensity(e14).
edge(e25).	intensity(e15).
edge(e26).	intensity(e18).
edge(e27).	intensity(e19).
	intensity(e21).
normal_constant_to_edge(r2,e7).	intensity(e22).
normal_constant_to_edge(r2,e8).	intensity(e23).

intensity(e24).

intensity(e25).

virtual(e5).

virtual(e9).

virtual(e11).

virtual(e12).

virtual(e13).

virtual(e17).

virtual(e20).

virtual(e26).

virtual(e27).

shadows(e4,e5).

shadows(e7,e12).

shadows(e8,e13).

shadows(e9,e11).

shadows(e15,e17).

l_connect s(e6,e7).

l_connects(e7,e6).

l_connects(e6,e8).

l_connects(e8,e6).

l_connects(e7,e10).

l_connects(e10,e7).

l_connects(e8,e10).

l_connects(e10,e8).

region(r1).

region(r2).

region(r3).

region(r4).

region(r5).

ground_plane(r5).

normal_changes_to_edge(r2,e6).

adjacent(r1,r2).

adjacent(r2,r1).

adjacent(r1,r5).

adjacent(r5,r1).

adjacent(r2,r4).

adjacent(r4,r2).

adjacent(r2,r5).

adjacent(r5,r2).

adjacent(r3,r5).

adjacent(r5,r3).

adjacent(r4,r5).

adjacent(r5,r4).

staggers(r1,r2).

staggers(r2,r1).

staggers(r1,r5).

staggers(r5,r1).

staggers(r2,r4).

staggers(r4,r2).

staggers(r2,r5).

staggers(r5,r2).

staggers(r3,r5).

staggers(r5,r3).

staggers(r4,r5).

staggers(r5,r4).

one_side_striped(e4,r1).

one_side_striped(e20,r1).

one_side_striped(e9,r2).

one_side_striped(e16,r3).

one_side_striped(e17,r4).

one_side_striped(e5,r5).

one_side_striped(e11,r5).

one_side_striped(e12,r5).

one_side_striped(e13,r5).

one_side_striped(e26,r5).

one_side_striped(e27,r5).

two_side_striped(e1,r1,r5).

two_side_striped(e2,r1,r5).

two_side_striped(e3,r1,r5).

two_side_striped(e4,r1,r5).

two_side_striped(e6,r4,r2).

two_side_striped(e6,r5,r2).

two_side_striped(e7,r5,r2).

two_side_striped(e8,r5,r2).

two_side_striped(e14,r3,r5).

two_side_striped(e18,r4,r5).

two_side_striped(e19,r4,r5).

upper_borders(e1,r1).

upper_borders(e2,r1).

upper_borders(e6,r2).

upper_borders(e7,r2).

upper_borders(e8,r2).

upper_borders(e14,r3).

upper_borders(e17,r4).

upper_borders(e18,r4).

upper_borders(e19,r4).

lower_borders(e3,r1).

lower_borders(e4,r1).

lower_borders(e6,r4).

lower_borders(e9,r2).

lower_borders(e16,r3).

t_junction(e1,e22).

t_junction(e2,e25).

t_junction(e4,e7).

t_junction(e3,e6).

t_junction(e18,e6).

t_junction(e18,e15).

t_junction(e19,e6).

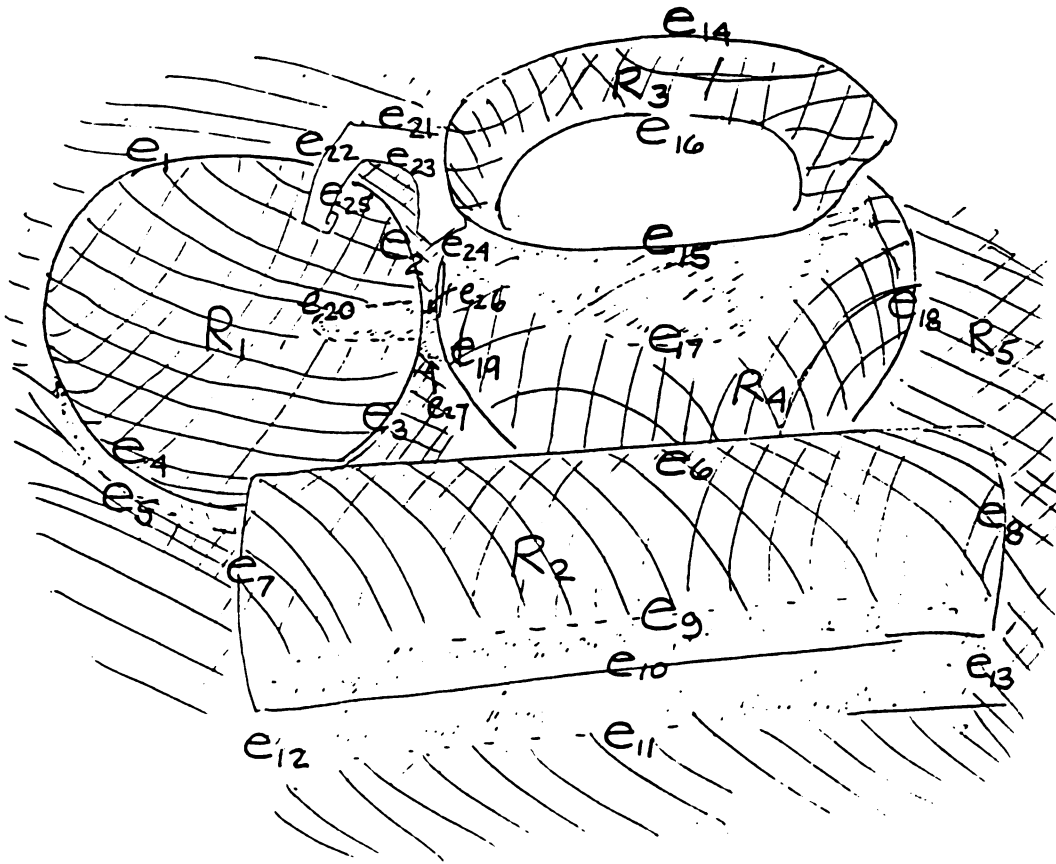
t_junction(e21,e14).

t_junction(e24,e15).

smoothly_connects(e1,e4).

smoothly_connects(e2,e3).

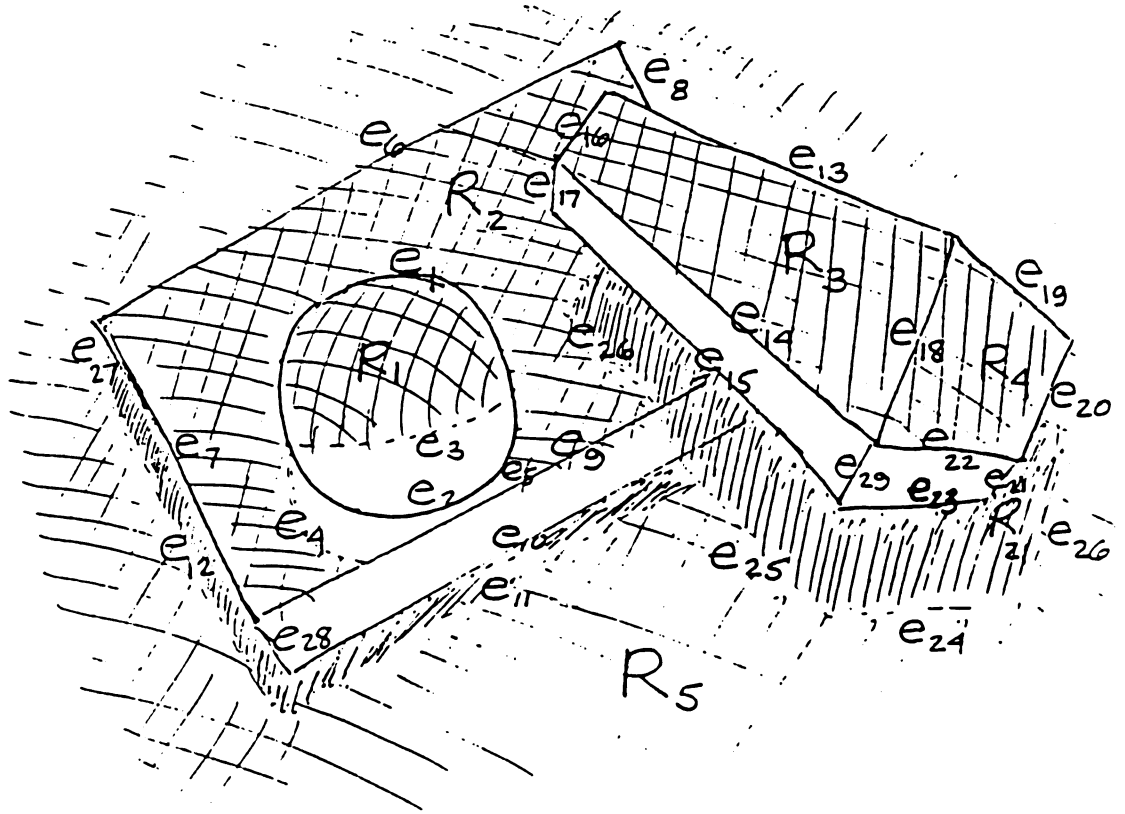
3. scene 1 — a tea pot, a plate, and a roll of paper



Labeling results:

set of blades	:	[e1,e14,e16,e2,e3,e4]
set of extremum	:	[e18,e19,e6]
set of P-shadows	:	[e11,e12,e13,e17,e5]
set of shadowing	:	[e15,e4,e7,e8,e9]
planar surfaces	:	[r5]
convex surfaces	:	[r2,r4]
concave surfaces	:	[r1,r3]
occluded regions	:	[r1,r4,r5]

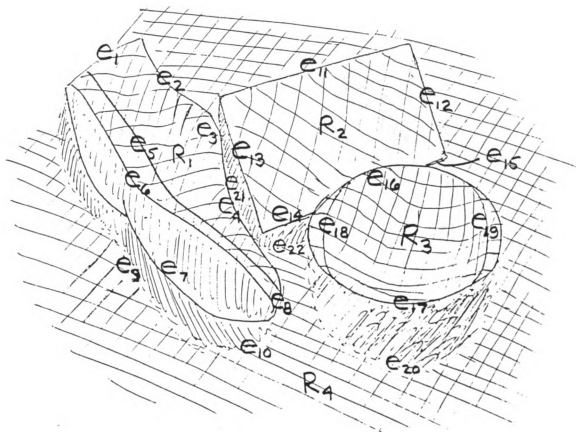
4. scene 2 — a book, an orange, and a metronome



Labeling results:

set of blades	:	[e13,e19,e20,e6,e8]
set of folds	:	[e14,e18,e22,e7,e9]
set of extremum	:	[e1]
set of P-shadows	:	[e11,e12,e24,e25,e26,e4,e5]
set of shadowing	:	[e14,e20,e22,e3,e7,e9]
planar surfaces	:	[r2,r3,r4,r5]
convex surfaces	:	[r1]
occluded regions	:	[r2,r5]

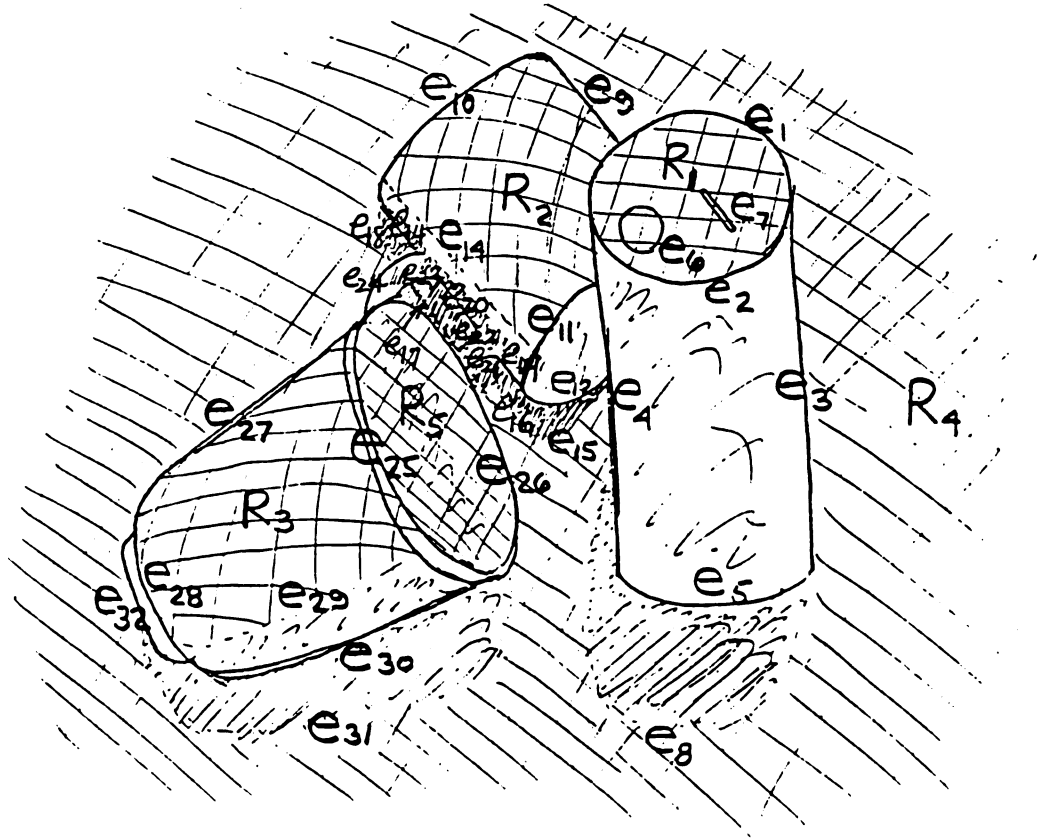
5. scene 3 — a shoe, a bowl, and box



Labeling results:

set of blades	:	[e11,e12,e13,e14,e15,e16,e17,e2]
set of folds	:	[e18,e19,e6]
set of extremum	:	[e1,e3,e4]
set of P-shadows	:	[e10,e20,e21,e22,e9]
set of shadowing	:	[e13,e14,e17,e6,e8]
set of marks	:	[e5]
planar surfaces	:	[r2,r4]
convex surfaces	:	[r1_1,r1_3]
concave surfaces	:	[r1_2,r3]
occluded regions	:	[r2,r4]

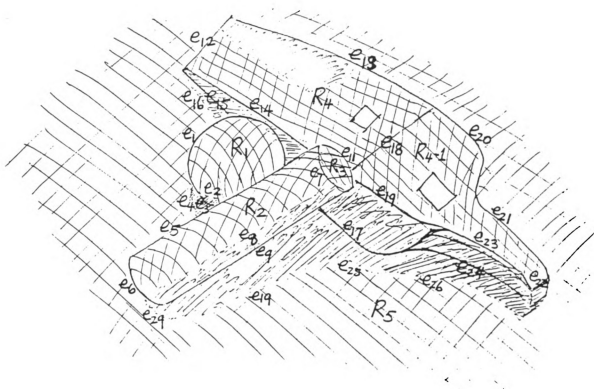
6. scene 4 — a can, a cup, and cup with cover



Labeling results:

set of blades	:	[e1,e10,e26]
set of folds	:	[e11,e2,e25]
set of extremum	:	[e27,e9]
set of P-shadows	:	[e15,e16,e17,e18,e31,e8]
set of shadowing	:	[e11,e13,e2,e29]
set of marks	:	[e6,e7]
planar surfaces	:	[r1,r4,r5]
convex surfaces	:	[r2,r3]

7. scene 5 — a vacuum cleaner, a can, and a tomato



Labeling results:

set of blades	:	[e12,e13,e19,e6]
set of folds	:	[e14,e18,e23,e7]
set of extremum	:	[e1,e20,e21,e22,e5]
set of P-shadows	:	[e10,e16,e25,e26,e27,e4]
set of shadowing	:	[e14,e19,e2,e23,e6,e8]
planar surfaces	:	[r3,r4,r5]
convex surfaces	:	[r1,r2,r4_1]
occluded regions	:	[r1,r4,r5]

BIBLIOGRAPHY

BIBLIOGRAPHY

- Agin, G. J. and Binford, T. O. "Computer Description of Curved Objects", *IEEE Trans. on Computer*, Vol. C-25, April 1976, 439-449.
- Agin, G. J. and Duda, R. O. "SRI Vision Research for Advanced Automation", in *Proceedings of the 2nd USA-Japan Computer Conference*, Tokyo, August, 1975, 113-117.
- Bajcsy, R. "Three-Dimensional Scene Analysis", *Proc. 5th ICPR*, Dec 1-4, 1980, Miami Beach, 1064-1074.
- Bajcsy, R. and Lieberman, L. "Texture Gradient as a Depth Cue", *Computer Graphics and Image Processing*, 5, 1976, 52-67.
- Ballard, D. H. and Brown, C. M. "Computer Vision", Prentice-Hall, Inc. 1982.
- Baumgart, B. G. "Winged Edge Polyhedron Representation", *Tech. Rep*, AIM-179, Comp. Sci. Dept., Stanford Univ., Stanford, Calif. 1972.
- Besl, P. and Jain, R. "Three-Dimensional Object Recognition", *Computer Surveys*, Vol. 17, No. 1, March 1985, 75-145.
- Besl, P. and Jain, R. "Range Image Understanding", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 19-23, 1985 San Francisco, CA, 430-449.
- Besl, P. and Jain, R. "Invariant Surface Characteristics for 3-D Object Recognition in Range Images", *Computer Vision, Graphics, and Image Processing*, 33, 1 (January), 1986, 33-80.

- Bhanu, B. and Ho, C. C. "CAD-Based 3D Object Representation for Robot Vision", *IEEE Computer*, Vol. 20, No. 8, August 1987, 19-35.
- Biederman, I. "Aspects and Extensions of a Theory of Human Image Understanding", *Computational Processes in Human Vision: An Interdisciplinary Perspective*, (Z. Pylyshyn Ed.), New York: Ablex, 1987.
- Birk, J. R., Kelley, R. B., and Martins, H. "An Orienting Robot for Feeding Workpieces Stored in Bins", *IEEE Trans. on Syst. Man Cybern.*, 11, 2 (Feb.), 1981, 151-160.
- Bolles, R. C. "Overview of Applications of Image Understanding to Industrial Automation", *Proc. of the SPIE Conference on Techniques and Applications of Image Understanding*, Washington. D.C., April, 1981, Vol. 281. SPIE, Bellingham, Wash., 134-140.
- Bolles, R. C. and Fischler, M. A., "A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data", *Proc. of the 7th IJCAI, Vancouver, August, 1981*, 637-643.
- Brady, M. "Computational Approaches to Image Understanding", *Computer Surveys*, Vol. 14, No. 1, March 1982, 3-71.
- Brooks, R. A., and Binford, T. O., "Geometric Modeling in Vision for Manufacturing", *Proc. of SPIE Conf. on Robot Vision*, Washington. D. C. August, 1981, Vol. 281, SPIE, 141-159.
- Charniack, E. and McDermott, D. "Introduction to Artificial Intelligence", Addison-Wesley, 1985.
- Chen, R. T. and Dyer, C. R. "Model-Based Recognition in Robot Vision", *ACM Computing Surveys*, Vol 18, No 1, March 1986, 66-108.

- Chen, S. W., "Computing a Pose Hypothesis from a Small Set of 3-D Object Features", *Computer Science Tech. Rep.*, 1986, Michigan State Univ.
- Clowes, M. B. "On Seeing Things", *Artif. Intell.* 2, 1971, pp79-112.
- Faig, W., "Calibration of Close-Range Photogrammetry Systems: Mathematical Formulation", *Photogrammetric Engineering and Remote Sensing*, Vol. 41, No. 12, 1975, 1479-1486.
- Faugeras, O. D. and Toscani, G. "The Calibration Problem for Stereo", *Proc. of IEEE Conf. on CVPR*, June 22-26, 1986, Miami Beach, pp15-20.
- Fukunaga, K, "Introduction to Statistical Pattern Recognition", *Academic Press*, New York, 1972.
- Gibson, J. J. "The Perception of the Visual World", *Boston, Houghton Mifflin Company*, 1950.
- Gleason, G. J., and Agin, G. J., "A Modular System for Sensor-Controlled Manipulation and Inspection, *Proc. of the 9th International Symposium on Industrial Robots*, Washington, D.C., March, 1979, 57-70.
- Goad, C., "Special-Purpose Automatic Programming for 3-D Model-Based Vision", *Proc. of the Image Understanding Workshop*, Arlington, VA. June, 1983, 94-104.
- Grimson, W., and Lozano-Perez, T., "Model-based Recognition and Localization from Tactile Data", *Proceedings of International Conference on Robotics*, Atlanta, GA, March 13-4, 1984.
- Hall, E. L., Tio, J. B. K., McPherson, C. A. and Sadjadi, F. A. "Curved Surface Measurement and Recognition for Robot Vision", *IEEE Workshop on Industrial Application of Machine Vision*, May 1-3, 1982, 187-199.

- Hambrick, L. N., Loew, M. H. and Corroll, R. L. "The Entry-Exit Method of Shadow Boundary Segmentation", *IEEE Trans. on PAMI*, Vol. PAMI-9, No. 5, Sept. 1987, 597-607.
- Henderson, T. C., "Efficient Segmentation Method for Range Data", *Proc. of the SPIE Conf. on Robot Vision*, Arlington, VA., May 1982, 46-47.
- Henderson, T. C., and Bhanu, B., "Three-Point Seed Method for the Extraction of Planar Faces from Range Data", *Conf. Record of the Workshop on Industrial Applications of Machine Vision*, Research Triangle Park, N.C., May, 1982, 181-186.
- Haralick, R. M., "Statistical and Structural Approaches to Texture", *Proc IEEE* 67 (1979), 786-804.
- Hoffman, R., and Jain, A. K., "Segmentation and Classification of Range Images", *IEEE Trans. on PAMI*, 9, 5 (Sept.), 1987, 608-620.
- Horn, B. K. "Understanding Image Intensities" *Artificial Intelligence*, 8 (1977), 201-231.
- Horn, B. K., "Obtaining Shape from Shading Information", in *The Psychology of Computer Vision* (P. H. Winston ed.), McGraw-Hill, 1985.
- Hu, G., Jain, A.K. and Stockman, G. "Shape from Light Stripe Texture", *Proc. of IEEE Conf. on CVPR*, Miami Beach, June 22-26, 1986, 412-414.
- Hu, G. and Stockman, G. "3-D Surface Solution Using Structured Light and Constraint Propagation", to appear on *IEEE PAMI*, 1988.
- Huffman, D. A. "Impossible Objects as Nonsense Sentences", in *Machine Intelligence*, vol. 6, Meltzer and Michie eds., Edinburgh Univ. Press, Edinburgh, U.K., 1971.

- Ito, M. and Ishii, A. "Range and Shape Measurement Using Three-View Stereo Analysis", *Proc. of IEEE Conf. on CVPR*, June 22-26, 1986, Miami Beach, pp14-19.
- Ittner, D. J. and Jain, A. K. "3-D Surface Discrimination from Local Curvature Measures", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, June 19-23, 1985, San Francisco, CA. 119-123.
- Jarvis, R. A. "A perspective on Range Finding Techniques for Computer Vision", *IEEE PAMI*, Vol. 5, No. 2, March 1983, pp122-139.
- Kanade, T. "A Theory of Origami World", *Artificial Intelligence* 13(1), 1980, 279-311.
- Kanade, T. "Recovery of the Three-Dimensional Shape of an Object from a Single View", *Artificial Intelligence*, 17 (1981), 409-460.
- Kender, J. "Shape from Texture", *TR CMU-C5-81-102*, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA., 1980.
- Koenderink, J. J. "What Does The Occluding Contour Tell Us About Solid Shape?", *Perception*, Vol. 13, 1984, 321-330.
- Koenderink, J. J., and vanDoorn, A. J., "The Internal Representation of Solid Shape with Respect to Vision", *Biol. Cybern.*, 32, 1979, 211-216.
- Lieberman, L., "Model-Driven Vision for Industrial Automation", *Advances in Digital Image Processing*, P. Stucki, Ed., Plenum, New York, 235-246.
- Laffey, T. J., Haralick, R. M. and Watson, L. T. "Topographic Classification of Digital Image Intensity Surfaces", *Proceedings of IEEE Workshop on Computer Vision : Representation and Control*, August 1982, 171-177.
- Le Moigne, J. and Waxman, A. M. "Multi-Resolution Grid Patterns for Building Range Maps", *U of Maryland, TR*.

- Levine, M. D., O'Handley, D. A. and Yagi, G. M. "Computer Determination of Depth Maps", *CGIP* Vol. 2, 1973, pp134-150.
- Lipschutz, M. "Differential Geometry", McGraw-Hill, 1969.
- Lowe, D. G. and Binford, T. O. "The Recovery of Three-Dimensional Structure for 3-D Machine Vision", *IEEE Trans on PAMI*, Vol 7, No. 3 May 1985, pp320-325.
- Malik, J., "Interpreting Line Drawings of Curved Objects", *International Journal of Computer Vision*, Vol. 1, No. 1, 1987, 73-103.
- Marr, D. "Vision", *Freeman & Company*, 1982.
- Marr, D. and Nishahara, H. K. "Representation and Recognition of the Spatial Organization of Three Dimensional Structure", *Proc. R. Soc. London B*, 200, 1978, pp269-294.
- Marr, D. and Poggio, T. "Cooperative Computation of Stereo Disparity", *Science*, 194, 1976, pp283-287.
- Medioni, G. and Nevatia, R. "Description of 3-D Surfaces Using Curvature Properties", *Proceedings of DARPA Image Understanding Workshop*, October 3-4 1984, New Orleans, LA. 291-299.
- Milgram, D. L., and Bjorklund, C. M., "Range Image Processing: Planar Surface Extraction", *Proc. of the 5th Intl. Conf. on Pattern Recog.*, Miami Beach, FL. Dec. 1980, 912-919.
- Millman, R. S. and Parker, G. D. "Elements of Differential Geometry", Prentice-Hall, Inc. 1977.
- Mishkin, M., Ungerleider, L. G. and Macko, K. A. "Object Vision and Spatial Vision : Two Cortical Pathways", *Trends in NeuroSciences*, 1983, 6, 414-417.

- Nevatia, R. and Binford, T. O. "Description and Recognition of Curved Objects", *Artificial Intelligence* 8 (1977), 77-98.
- Oshima, M. and Shirai, Y., "Object Recognition Using Three-Dimensional Information", *IEEE Trans. on PAMI*, 5, 4 (July), 1983, 353-361.
- Perkins, W. A., "A Model-Based Vision System for Industrial Parts", *IEEE Trans. on Computers*, 27, 2 (Feb.), 1978, 126-143.
- Persoon, E., and Fu, K. S., "Shape Discrimination Using Fourier Descriptors", *IEEE Trans. on Syst. Man and Cybern.*, 7, 3 (March), 1977, 170-179.
- Pietikainen, M., and Harwo D. "Depth from Three Camera Stereo", *Proc. of IEEE Conf. on CVPR*, June 22-26, 1986, Miami Beach, pp2-8.
- Ponce, J. and Brady, M. "Toward a Surface Primal Sketch", *Proceedings of IEEE International Conference on Robotics and Automation*, St. Louis, Mo., March 22-28 1985, 420-425.
- Popplestone, R. J., Brown, C. M., Ambler, A. P., and Crawford, G. F., "Forming Models of Planar-and-Cylinder Faceted Bodies from Light Stripes", *Proc. of the 4th IJCAI*, Tbilisi, USSR, Sept. 1975, 664-668.
- Posdamer, J. L. and Altschuler, M. D. "Surface Measurement by Space-encoded Projected Beam Systems", *CGIP*, 18, 1982, pp1-17.
- Rocker, F. and Kiessling, A. "Methods for Analyzing Three Dimensional Scenes", *Proc. of 4th IJCAI*, 1975, 669-672.
- Rosenfeld, A. and Kak, A. C. "Digital Picture Processing" 2nd Ed., *Academic Press*, 1982.
- Sekuler, R. and Blake, R. "Perception", *Alfred A. Knopf, Inc.*,
- Shirai, Y. "Recognition of Polyhedrons with a Range Finder", *Artificial Intelligence*, Vol 4, No 3, Oct. 1972, pp243-250. 1985.

- Shirai, Y., "Recognition of Real-World Objects Using Edge Cues", *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds., Academic Press, New York, 1978, 353-362.
- Shneier, M., "Models and Strategies for Matching in Industrial Vision", *Tech. Rep. TR-1073*, Comp. Sci. Dept., Univ. of Maryland, College Park, July, 1981.
- Shrikhande, N. and Stockman, G. "Surface Orientation from Striped Lighting", *Proc. of 5th Scandinavian Conf. on Image Analysis*, June 1987, Stockholm, Sweden.
- Stevens, K. A. "The Visual Interpretation of Surface Contours", *Artificial Intelligence* 17 (1981) 47-73.
- Stockman, G. C., Kopstein, K., and Benett, S., "Matching Images to Models for Registration and Object Detection via Clustering", *IEEE Trans. on PAMI*, 4, 3 (May), 1982, 229-241.
- Stockman, G. C., and Chen, S. W., "Detecting the Pose of Rigid Objects: A Comparison of Paradigms", *Proc. SPIE Conf. on Electro-Optic Imaging Systems and Devices*, Los Angeles, CA. 11-16 Jan., 1987.
- Stockman, G. C., Chen, S. W., Hu, G. and Shrikhande, N. "Recognition of Rigid Objects Using Structured Light" *Proc of IEEE Conf. on Systems, Men and Cybernetics*, Oct. 1987, Washington D. C.
- Stockman, G. C., "Object Recognition and Localization via Pose Clustering", *Computer Vision, Graphics, and Image Processing*, 40, 1987, 361-387.
- Strand, T.C. "Optical Three-Dimensional Sensing for Machine Vision", *Opt. Eng.*, 24(1), 33, 1985.

- Tsai, R. Y. "Multiframe Image Point Matching and 3-D Surface Recognition", *IEEE Trans on PAMI*, Vol 5, No. 2, March 1983, pp159-173.
- Tsai, R. Y. "An Efficient and Accurate Camera Calibration Technique for 3-D Machine Vision", *Proc. of IEEE Conf. on CVPR*, June 22-26, 1986, Miami Beach, pp364-374.
- Wallach, H. and Floor, L., "The Use of Size Matching to Demonstrate the Effectiveness of Accommodation and Convergence as Cues for Distance", *Perception and Psychophysics*, 10, 1971, 423-428.
- Waltz, D. I. "Generating Semantic Description from Drawings of Polyhedral Scenes with Shadows", in *The Psychology of Computer Vision* (Winston, P., Ed.), McGraw-Hill, NY. 1975.
- Wang, Y.F., Mitiche, A. and Aggarwal, J.K. "Computation of Surface Orientation and Structure of Objects Using Grid Coding", *IEEE Trans. on PAMI*, Vol. PAMI-9, No. 1 1987, pp129-137.
- Will, P. M. and Pennington, K. S. "Grid Coding : A Preprocessing Technique for Robot and Machine Vision", *2nd IJCAI*, Imperial College, London, 1971, pp66-70.
- Witkin, A. P., "Recovering Surface Shape and Orientation from Texture", *Artif. Intell.*, 17, 1(Aug.), 1981, 17-47.
- Woodham, R. J., "Photometric Stereo: A Reflectance Map Technique for Determining Surface Orientation from Image Intensity", *Proc. of the SPIE Conf. on Image Understanding Systems and Industrial Applications*, San Diego, Calif., August, 1978, Vol. 155, SPIE, 136-143.
- Yakimovsky, Y. and Cunningham, R., "A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras, *Computer Graphics and Image Processing*, 7, 1978, 195-210.

Yang, H. S., Boyer, K. L. and Kak, A. C. "Range Data Extraction and Interpretation by Structured Light", *TR 1984*, pp199-205.

Yang, H. S. and Kak, A. C. "Determinating of the Identity, Position and Orientation of the Topmost Object in a Pile", *CVGIP*, 36, 1986, 229-255.

MICHIGAN STATE UNIV. LIBRARIES



31293108120803