Shape From Textures:
A Paradigm for Fusing Middle Level Vision Cues

Mark Laurence Moerdler

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY
1989

ABSTRACT

Shape From Textures: A Paradigm for Fusing Middle Level Vision Cues

Mark Laurence Moerdler

This research proposes a new approach to the problem of deriving the orientation, segmentation, and classification of surfaces based on **multiple** independent textual cues. The generality of this approach is due to the interaction between textural cues, thus allowing it to extract shape information from a wider range of textured surfaces than any individual method. The method consists of three major phases: the calculation of orientation constraints for sub-image elements called "texel patches", the consolidation of constraints into a "most likely" orientation per patch, and finally the reconstruction of the surface.

During the first phase, the different shape-from-texture components generate *augmented texels.* Each augmented texel consists of the 2-D description of a texel patch and a list of weighted constraints on its orientation. The orientation constraints for each patch are potentially inconsistent or potentially incorrect because the shape-from methods are applied to noisy images, locally based, and derive constraints without a priori knowledge of the type of texture or number of surfaces. The constraints are weighted by each shape-from method based on an intra-cue correctness factor. This factor attempts to measure how closely the constraint fulfill the underlying assumptions of the cue. The orientation constraints' weights are then normalized between cues in order to assure that no cue predominates unfairly.

In the second phase, all the orientation constraints for each augmented texel are consolidated into a single "most likely" orientation by a Hough-like transformation on a tesselated Gaussian sphere. The system iteratively re-

analyzes each of the texel patches, calculating the "most likely" orientations for each patch.

Finally, the system re-analyzes the orientation constraints to determine which augmented texels are part of the same constraint family and which cues were used to generated the valid constraints. In effect, this both segments the image into regions of similar orientation and supplies texture classification information.

The robustness of this approach is illustrated by a system that fuses the orientation constraints of five shape-from cues and solves real camera-acquired imagery.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I am deeply indebted to my adviser, John R Kender, for his patience, insights, and guidance throughout our long relationship. Without his great dedication this work would not be possible.

I would like to thank the members of my dissertation committee: Peter Allen, Theodore Bashkow, Terrance Boult, and Stephen Unger. Their comments and helpful questions have helped to improve this work.

A special thanks must be given to Arnow Penzias for words of encouragement and help without which I would never have started along the path to a Ph.D.

It is said that 'A tree is only as strong as its roots'. My roots and support arise from my family: my grandparents, parents, in-laws, siblings, and especially my wife all of whom made this possible. Thank you.

Finally, to my wife a heartfelt thanks. For the years of patience, encouragement, and understanding. I can never completely express my appreciation.

1. Introduction

Determining the three dimensional representation of the surfaces in an image is an important step in the image recognition process. One of the major approaches to this problem centers on the use of texture-based cues which arise in almost all natural and synthetic images [Gibson 50]. Surfaces such as brick walls, fabrics, and tree bark are classic examples of textured surfaces.

Texture cues can be used to solve the orientation of textured surfaces because a change in the orientation of a textured surface translates into a deformation in the texturing of the perceived surface. For example, the surface of figure 1-1 consists of uniformly spaced and sized circles. When this surface is seen from an angle as in figure 1-2 deformation occurs in the perceived spacing of these circles, as well as in the shape of the circles. By utilizing these and other types of perspective effects (for example, foreshortening compression and size distortion), and making assumptions about the surface and its texture, the orientation of textured surfaces could be recovered. One contribution of this work is that it makes explicit these and other underlying assumptions.

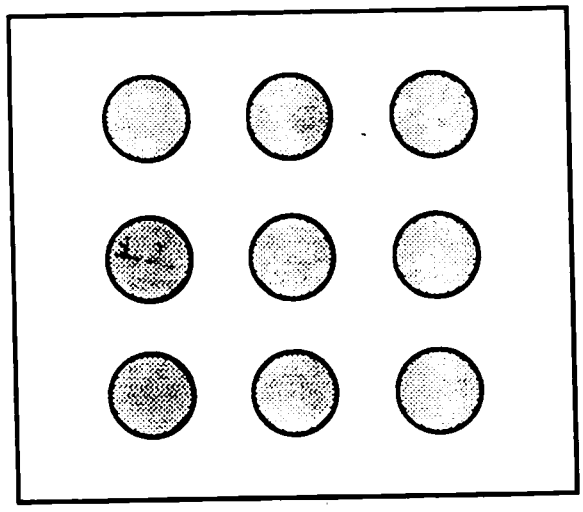


Figure 1-1: A surface textured with uniformly sized and spaced circles.

Shape-from-texture methods are based on measuring deformations that

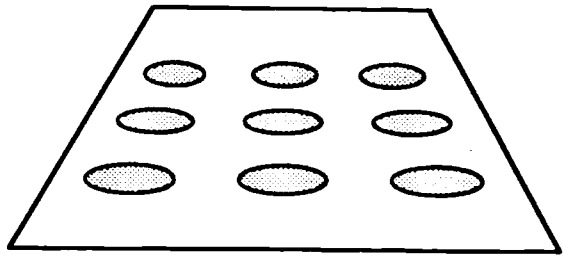


Figure 1-2: The surface of Figure 1-1 seen under perspective

occur when a textured surface is viewed under perspective projection. This perspective distortion is imaged with a change in some aspect of the texture. In order to simplify the recovery of the orientation parameters from this deformation, researchers have imposed limitations on the applicable class of textured surfaces. Some of the limiting assumptions include uniform texel[1] spacing [Kender 80; Kender 83; Moerdler and Kender 85], uniform texel size [Ikeuchi 80a; Ohta et. al. 81; Aloimonos and Swain 85], uniform texel density [Aloimonos 86], and texel isotropy [Witkin 80; Davis, Janos and Dunn 83; Dunn 84]. Each of the above listed assumptions are strong limitations, causing the methods based on them to be appliable to only a limited range of real images. A solution to these problems would to be create a general shape from texture system containing multiple shape from texture cues that, fused together, solves images that the individual cues could not solve.

1.1 Fusion Models - Heterogeneous vs. Homogeneous

However, prior to defining a paradigm for fusing multiple corroborating and conflicting cues an important distinction must be made in how constraints are integrated. Two different types of fusion models have been used in previous

---

[1] A textured surface patch is defined as a *texel*.

research, which we call heterogeneous and homogeneous fusion. Until now this difference between the models has not been articulated.

> **Definition 1-1:** We define *heterogeneous fusion* as the combining of different cues which utilize either different data, or the same raw data at different levels of abstraction, to generate related information.
>
> **Definition 1-2:** We define *homogeneous fusion* as the combining of different cues which utilize the same raw data at the same level of abstraction to generate the same type of information.

In general knowledge fusion systems in computer vision have performed heterogeneous fusion (e.g. [Spatial Reasoning and Multi-Sensor Fusion 87; Bixler and Miller 87]) rather than homogeneous fusion. These systems augmented the visual data generated by one cue, such as stereo or shading, with an overlapping set of data from additional cues. They attempt to combine cues in order to "collect" all of the world knowledge necessary for a task.

In this research the problem of homogeneous fusion is considered, and a paradigm for integrating a diverse set of corroborating and conflicting cues is proposed. The paradigm attempts to recover correct data even when the input is noisy, or when many of the cues are erroneous.

1.2 The General Fusion Paradigm

Now we will present the theory of how multiple cues are integrated. Specifically, this work proposes a new approach to the problem of deriving the orientation and segmentation of surfaces based on *multiple* independent cues. The generality of this approach is due to the cumulative interaction between cues (a form of synergistic fusion) rather than the power of any specific cue. The paradigm consists of three major components:

- The subdivision of cues into groups of homogeneous cues.
- The choosing of the level of abstraction.
- The creation of a constraint weighting scheme.

For the first step the cues can be grouped into classes by means of their "area of support" in the image(s). Different shape-from cues utilize different groupings of image areas, e.g. shape-from-stereo cues match image areas, lines, or points between different images [Moerdler and Boult 88] while shape-from-man-made-textures measure changes between "texels" (however they are defined) in the image. If the shape-from methods utilize the same scale and type of information, then all of the constraints that relate to the same image locality can be integrated to generate a single surface element description for that image locality. However, if the methods utilize different scale and/or different data types, then a common but more abstract level must be chosen.

The second step chooses the level of abstraction at which to fuse the cues. The level should depend on the image areas that the cues utilize in generating their constraints(e.g. fuse constraints at each point if the constraints are generated at each point). If too low a level is chosen, then the fusion step will have to fuse a handful of constraints for each of a very large number of image localities. At the other extreme, if the level of abstraction is too high (e.g. fusing all of the constraints for the image when the constraints are generated for smaller surface areas), then image information becomes too sparse, the surface is generated by information from only a few localities, and may be coarsely recovered.

The last step in the paradigm, usually made up of multiple components, is the creation of a constraint weighting scheme. The weighting scheme proposed here consists of two components: an intra-cue correctness factor, and an inter-cue normalization factor. The intra-cue correctness factor attempts to model how closely each specific constraint fits the underlying assumptions of the cue, and therefore how "good" the constraint is. But since the cues generate multiple, possibly inconsistent constraints per image locality, it is also necessary to weight the constraints generated by different cues. This normalization factor attempts to assure that no single cue gains precedence over other cues simply because its

form creates more constraints. The two components work together to assure that the combination of multiple consistent cues win out in determining a surface orientation.

1.3 Fusing Multiple Shape-from-texture methods

This work proposes a new approach to this problem of deriving the orientation and segmentation of surfaces based on *multiple* independent textual cues based on the fusion paradigm described above. The method, as diagrammed in figure 1-3, consists of three major phases: the calculation of orientation constraints and the generation of *texel patches*[2] (here, the proper level of abstraction), the consolidation of constraints into a "most likely" orientation per patch, and finally the reconstruction of the surface by standard integration methods.

During the first phase, the different shape-from-texture components generate texel patches and *augmented texels*. Each augmented texel consists of the 2-D description of a texel patch and a list of weighted constraints on its orientation. The orientation constraints for each patch are potentially inconsistent. This occurs due to many reasons including that:

- the shape-from methods may be applied to noisy real images;
- they are locally based;
- they derive constraints without a priori knowledge of the type of texture or number of surfaces.

In the second phase, all the orientation constraints for each augmented texel are consolidated into a single "most likely" orientation by a Hough-like accumulation over a tesselated Gaussian sphere. During this phase the system will also logically merge together all the augmented texels that cover the same locality ("surface patch") of the image. This is possible since many of the shape-

---

[2]A *texel patch* is a 2-D description of a sub-image that contains one or more textural elements. The number of elements that compose a patch is dependent on the specific shape-from-texture method.

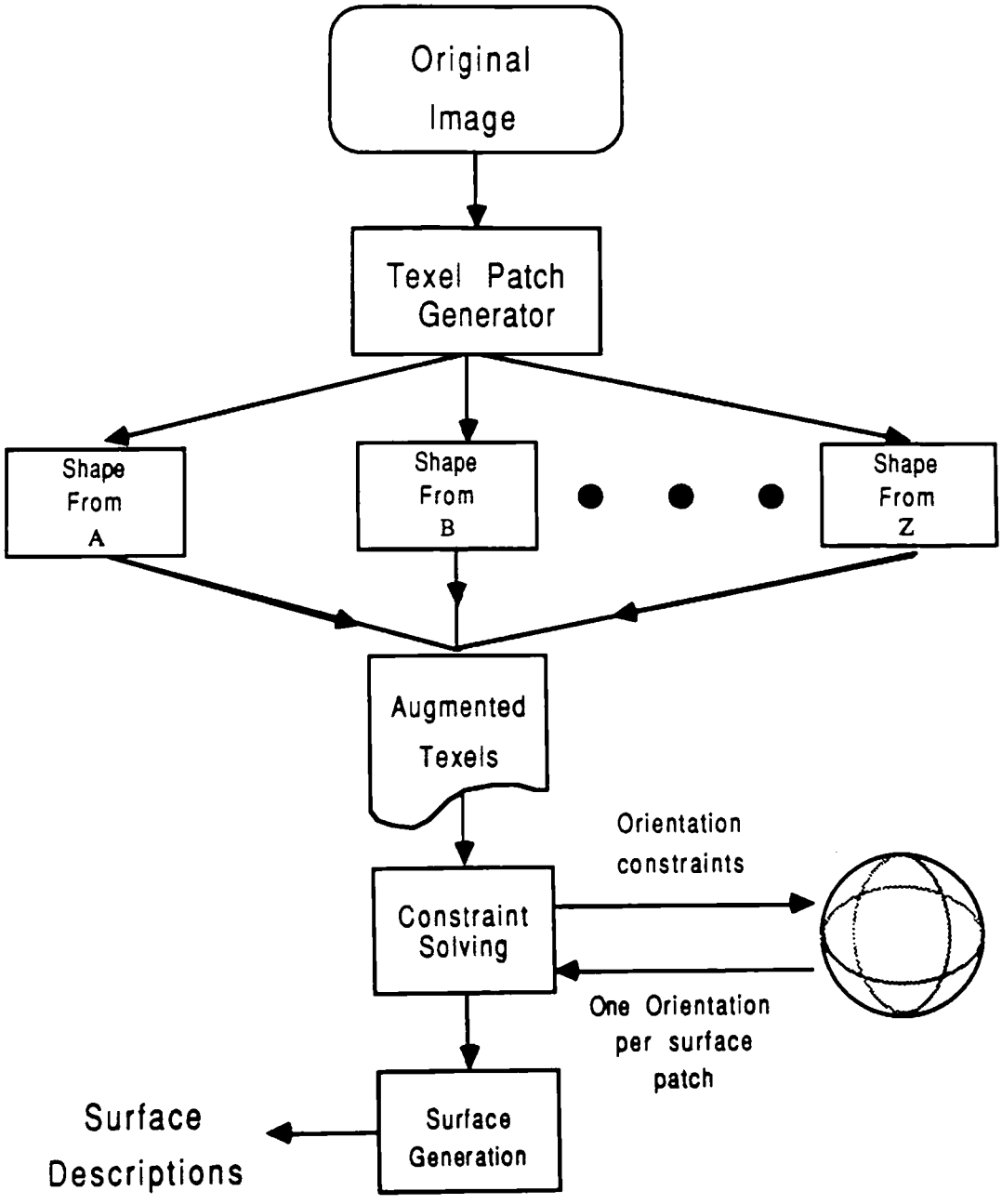


Figure 1-3: Integrating multiple shape-from methods

from components define "texel" similarly, thus the constraints generated should be merged, and a single orientation generated for the surface patch. The orientation constraints' weights are normalized between shape-from methods in order to assure that no method predominates unfairly: some shape-from-texture modes "naturally" generate a larger number of constraints per texel patch (e.g.

shape-from-uniform-texel-spacing generates more constraints than shape-from-uniform-texel-size).

In those instances in which multiple "most likely" orientations are found, the system checks the validity of each of the texel's constraints, as measured by its spatial and orientation relationship to neighboring texels. If a constraint was generated in a calculation involving other texels, but the constraint is not consistent with those other texels, then the constraint is not considered as valid for this texel. This allows the system to prune out some of the constraints; after representing the constraints as curves on the tesselated Gaussian sphere, it determines a single "most likely" orientation. In short, this is a type of relaxation applied to the "intrinsic image" of surface orientation.

Finally, the system re-examines, for each orientation constraint, those augmented texels that are part of the same constraint family and groups them together by surface orientation. In effect, this segments the image into regions of similar orientation. The surface is then reconstructed from these surface patches be means of standard surface reconstructions, for example ones based on generalized smoothing spline functions.

1.4 Contribution of This Research

This work is new in ten ways:

1. The system has been applied to a wide range of real imagery. The images are real world images in which few a priori assumptions are made about the number of surfaces in the image, the type of texture (within a fairly wide class), the placement of textural elements, or the curvature of the surface(s).
2. The paradigm allows for the integration of cues, some of which are statistical while others are structural. This "blurring" of the statistical/structural boundary increases the overall applicability of the paradigm.
3. Existing shape-from-texture cues have been analyzed and found to be composed of two major constituent components: their definition of a texel, and their specific orientation recovery algorithm. This separation simplifies the process of defining a range of new shape-

from-texture cues. New cues can be created by taking any of the texel definitions and combining it with any of the recovery algorithms.

4. A robust shape-from-texture system has been built and demonstrated, that is based upon the integration of multiple methodologies that cooperate and/or conflict.
5. Because the shape-from-texture system solves the orientation of each texel patch independently it is able to restrict the effects of noise to the specific texels that are either generated by noise or directly affected by the noise.
6. A new data structure, the augmented texel, was created in order to simplify the fusion of multiple orientation constraints. Furthermore, this data structure allows for a simple description of many of the salient features of surface patches.
7. A two level constraint weighting scheme has been created that allows the integration of orientation constraints derived by different methodologies, even under conditions in which one or more of the methods generate noisy or incorrect constraints.
8. Since the method fuses constraints per texel patch the method is shown to aid in the segmentation of surfaces.
9. The method aids in the separation of transparent surfaces.
10. Once all of the texel patch orientations are recovered the system can determine which cues were applicable to the surfaces, and thus aid in the classification of the type of texture on the surface(s): fusion leads to segmentation and recognition.

1.5 Assumptions and Limitations

Previous research in the use of texture to derive shape, which will be described in Chapter 2, was based upon underlying assumptions about either the texture and/or number of surfaces. This research has attempted to relax and remove some of these underlying assumptions. The multiple shape-from-texture system described herein moves the state of the art towards the ideal of no prior knowledge about the number of surfaces in the image, the planarity or non-planarity of the surface(s), the type of texture, or the placement of texels.

This is possible because the system models and controls the specific shape-from-texture modules separately from each other. The modules are based

upon assumptions that limit the class of applicable textures; the approximateness with which the texture(s) in the image meet the assumptions is utilized in the generation of each constraint's weighting. This decouples the assumptions from the orientation constraints generated, and thus allows a more powerful system to be built based upon *multiple* shape-from-texture methods.

1.6 Organization of Subsequent Chapters

The remainder of this work will describe the particular components of the knowledge fusion approach described above and demonstrate its effectiveness.

In Chapter 2 we review the state of the art in shape-from-texture. A quick summary is given of each of the existing shape-from-texture methods and a unifying formalism is supplied that not only simplifies fusion it defines how additional shape-from-texture cues can be created.

In Chapter 3 the levels at which orientation information can be fused is discussed and the surface patch level is chosen and justified. In order to simplify the fusion of surface-patch-level orientation constraints, a new data structure, the *augmented texel*, is defined. Many of the shape-from cues discussed in Chapter 2 are then reinterpreted in this uniform framework that allows these cues to cooperate.

Chapter 4 describes a bin based constraint fusion method utilizing a tesselated Gaussian sphere. The method fuses orientation constraints, ignores incorrect constraints, and generates surface orientation information.

In Chapter 5, experimental results on real and synthetic data are presented that show the feasibility of this approach, including but also going beyond the fusing of shape-from methods.

We show in Chapter 6 how these methods aid in the separation and segmentation of surfaces, including transparent overlapping surfaces; further, these methods are shown to aid in the classification of the type of texture on the

surface.

The concluding chapter, Chapter 7, summarizes the methodology applied in this work.

2. History of the Use of Texture to Derive Shape

This chapter will describe previous research into the use of texture to derive the shape of textured surfaces. This research has covered a span of approximately 40 years and has led to the development of a number of shape-from-texture cues.

The recovery of orientation by means of texture is a hard problem that has not been solved without utilizing initial assumptions about either the surface(s) or the texture. Previous shape-from-texture methods are based on fairly stringent assumptions. Since these assumptions limit the classes of textured surfaces that the method is applicable to, these methods are best categorized according to the most limiting assumption -- the class of textures they are applicable to.

The remainder of this chapter surveys the most common simplifying assumptions that have been used to generate shape-from-texture methods. For each assumption the major shape-from-texture methods based on it are described and their limitations discussed. Since this classification method is based upon the major textural assumptions that are used, the ordering will not necessarily follow the chronological order in which the methods were created.

This chapter will briefly describe the basic formulation of each of the major shape-from-texture methods. Chapter 3 will elaborate on the specific shape-from-texture methods that were incorporated in the multiple shape-from-texture system described in this work. We will also specify there exactly how the method was implemented, and how its implementation differs from the previous implementations of these shape-from-texture methods we now discuss.

2.1 General Approaches to Shape-From-Texture

The use of texture to recover the shape of surfaces appears to have been first studied extensively by the psychologist Gibson [Gibson 50]. He theorized that under the assumption that the textural primitives could be identified and "counted", the gradient of textural density will define the orientation of a plane

(often the ground plane). The limiting factor in this theory is whether the textural elements can be identified and counted.

One of the more important contributions of his work is the description of how a change in the orientation of a textured surface translates into a distortion in the perceived surface. He divided perspective distortion into its two constituent parts:

- *Scaling Distortion:* The size of and spacing between textural primitives decreases as the surface recedes from the viewer. This type of distortion creates a scaling change as shown in figure 2-1.
- *Foreshortening Compression:* Textural primitives are themselves compressed in the direction of surface inclination. If a textural primitive is sufficiently large then the part of the primitive that is closest to the viewer will be proportionally larger than the part that is farther away. (See figure 2-2).

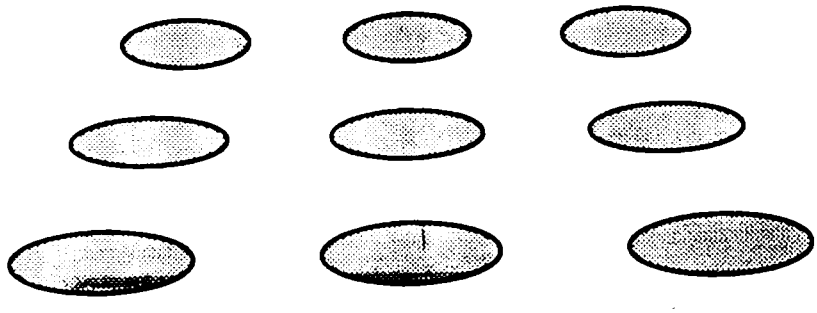


Figure 2-1: The effects of size distortion
on a perfectly regular spacing of uniform sized spots

Gibson's work was psychologically anecdotal rather than rigorously mathematical; he never defined the equations that describe perspective distortion. Rather, his work laid the foundation for the mathematically based methods that will be described.

Between Gibson's work in 1950 and the mid 1970's [Bajcsy and Lieberman 76] there is a large gap in shape-from-texture research with few references to the use of texture to derive shape. One of the few exceptions, that of Horn's paper

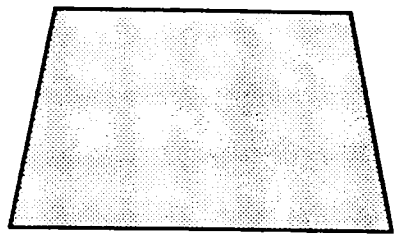


Figure 2-2: The effects of foreshortening compression on a square

on shape from shading [Horn 70], noted the relationship between shape-from-texture and shape-from-shading, but with the cameras available at the time, he was unable to gather accurate enough data to compute anything.

2.1.1 Basic Assumptions Defined by Stevens

Stevens [Stevens 79] built upon Gibson's work by attempting to describe those assumptions that are both sufficient and necessary to derive surface parameters from texture. He gave a categorization of the types of textural cues and texture effects that are due to perspective distortion. His results were psychologically oriented, and insufficient in themselves to account for known human capabilities to derive shape-from-texture, nor did they translate into shape-from algorithms. Moreover, the validity of his assumptions were not experimentally proved.

The rest of the approaches that will be described are based on more specific assumptions about the surface and/or the texture of the surface than the work of Gibson or Stevens, producing realizable and experimentally testable algorithms. All of these algorithms are based on the following important assumptions:

- Natural texture does not mimic projective effects, nor does it cancel those effects out (by this they mean: the surface texture is, in some sense, statistically stationary).
- Surface patches are locally planar.

2.1.2 Kender's Generalized Paradigm For Shape-From-Texture

Kender [Kender 80] describes a generalized paradigm for creating shape-from-texture methods. The paradigm is based on a conceptual and representational tool, the Normalized Textural Property Map (NTPM) which "de-projects" the effects that surface orientation has on primitive textural properties. The shape-from-texture paradigm consists of three main steps [Kender 80, page 81]:

1. The creation of the Normalized Texture Property Map. In this step a surface orientation representation is chosen; the camera parametization including texel position in the image is calculated; a texture property (e.g slope, area, length) is selected; and assumptions are made about the preferred constituent-to-surface relationship (e.g. are the texels located on the surface).
2. Two NTPMs with the same texture property are chosen. Based on the assumption that the two NTPMs share a microplane, a constraint curve in the surface orientation space is generated.
3. Two constraint curves sharing a microplane are selected to derive one or more surface orientations. The number of orientations generated are based on the texture property as well as whether orthographic or perspective distortion is used to derive the constraints.

Extended versions of the paradigm are also given which allow it to deal with non-planar surfaces as well as multiple surfaces.

Kender lists a number of texture properties that can be used to create shape-from-texture methods. These include texel slope, length of a major axis of elongation of a texel, image angle, texel area, texel density, eccentricity, and skewed symmetry. He does not define the specific equations necessary to derive surface orientation for many of these texel properties; rather, he supplies a general paradigm which can and has been used to create shape-from-texture

methods.

2.2 Uniform Texel Spacing

Kender's shape-from-uniform-texel-spacing cue [Kender 80; Kender 83] utilizes the NTPM paradigm to create a shape-from-texture method that recovers surface constraints from the apparent change in spacing between textural primitives that are assumed to be evenly spaced in the scene. These textural primitives, since they are spacings rather than lengths, are virtual textural elements. From a viewpoint normal to the surface, these spacings can be considered to be the equally-spaced intersections of actual textural elements with an arbitrary virtual line. When the surface is viewed in perspective to the viewer, the spacing between virtual primitives changes in an orderly way. This method uses the detection of a change in texel spacing to calculate local surface orientation by finding an equivalent representation of local surface gradient: the local vanishing line of perspective[3]. The slope and position of the vanishing line uniquely define the orientation component of the 3-dimensional representation of the surface. The 3-dimensional position of the surface can not be determined purely from single image textural information.

The derivation of surface constraints from textural cues can be performed by back-projecting the textural primitives onto a hypothesized local surface as can be seen graphically in figure 2-3. Mathematically, the vanishing points are generated by means of the back-projection formula relating texel position and the slopes of the constructed lines. Given any two texels T_1 and T_2 whose relative positions are P_1 and P_2, if the distance from T_1 to the mid-texel is equal to L and the distance from T_2 to the same mid-texel is equal to R, the vanishing point distance is given by :

---

[3]The vanishing line is a representation for surface orientation that is mathematically equivalent to any of the representations given in [Shafer, Kanade, and Kender 83].

$$\frac{X - P_1}{L} = \frac{X - P_2}{R}$$

where X is the vanishing point distance. Rewriting the equation we have:

$$X = \frac{L \times P_1 - R \times P_2}{L - R} \qquad (2\text{–}1)$$

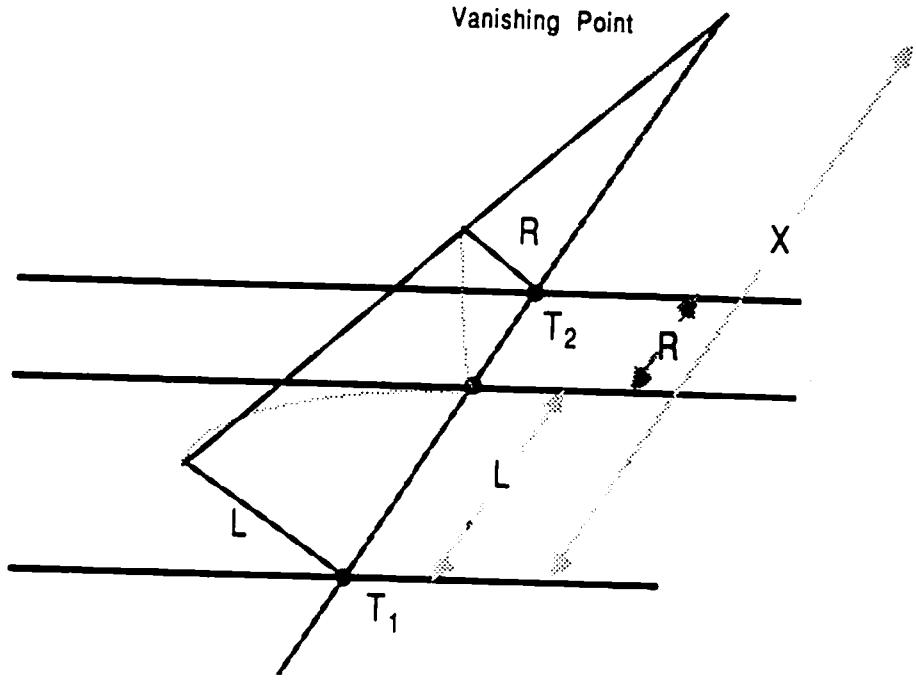


Figure 2-3: A geometrical representation of back-projecting.

Using the image position of three or more textural elements and the hypothesized surface parameters, this approach is able to recover constraints on surface orientation, under the assumption that the back-projected features are regularly distributed in real space and separable. The method has been tested on both synthetic noisy and real noisy images [Moerdler and Kender 85].

2.3 Uniform Texel Size

Another approach that derives the shape of surfaces assumes that all of the texels are of the same size prior to the perspective effect. As the orientation of the textured surface increases, the size of imaged texels decreases. This simplification is valid as long as the texel size is small relative to the surface orientation otherwise foreshortening compression will shrink the part of the texel farther away from the viewer.

At least four different algorithms based on this assumption have been created: Ikeuchi's [Ikeuchi 80a], Ohta's [Ohta et. al. 81], Aloimonos' [Aloimonos and Swain 85], and Blostein's [Blostein 87]. Each method utilizes the same basic assumption of uniform texel size in a different way, and in combination with other limiting assumptions.

Ikeuchi's method requires not only uniform size but also a priori knowledge of the texels to derive the shape of non-planar surfaces; Ohta's approach assumes only uniform size but is limited to planar surfaces only; Aloimonos' method combines the previous two approaches to reconstruct planar and non-planar surfaces; and lastly, Blostein's method derives the orientation of planar naturally-textured surfaces from the change in the size of texels as approximated by disks.

2.3.1 Uniform Texel Size with A Priori Knowledge

Ikeuchi's [Ikeuchi 80a] approach is based on the existence of a priori knowledge of the shape of the specific textural primitives, called generators, that cover all the surfaces in the image. In addition to this strong assumption (a priori knowledge of the shape of the texels) Ikeuchi has the following weaker assumptions:

1. The surface is covered by a uniform texture comprised of a single type of texel repeated across the surface.
2. The texels are small in comparison to the distance between the surface and the viewer. This assumption allows Ikeuchi to replace

the perspective projection by a kind of spherical orthographic projection.

3. Each texel is small in comparison to the change in the surface orientation. Thus the surface patches can be considered as locally planar.

Using the generators, Ikeuchi's method is able to recover one constraint on the orientation of *each* individual texel in the image. Yet, as stated earlier, two constraints are necessary to uniquely define the orientation of any surface patch. Therefore, Ikeuchi proposed two solutions to recovering the second constraint: either three images could be taken from three different perspectives[4], or an iterative propagation method can be used that constrains the orientation of texels from the occluding boundaries inward. This constraint propagation method is based on Ikeuchi's shape-from-shading algorithms [Ikeuchi 80b; Ikeuchi and Horn 81] . Under the assumption that the surface is "smooth", this algorithm is able to constrain the orientation of the surface.

The surface is considered to be comprised of patches where each patch (i,j) contains one texel and an orientation in terms of *F* and *G*. *F* and *G* are stereographic representations of orientation. This is mathematically equivalent to the more common gradient space parameters *p* and *q*, or to vanishing lines. At each iteration of the method a new *F* and *G* is calculated, from two terms. The first is the average of the previous *F*s and *G*s in its neighborhood The second is a heuristically generated weighting factor ($\frac{\lambda}{16}$) times the difference between R (the function that determines the ideal distortion from the stereographic gradient map) and I (the actual texture distortion "reflectance"). This last difference is an error adjustment term at the point (i,j), obtained from comparing the perceived texel and the value generated from *R*, *F*, and *G*.

The weighting factor $\frac{\lambda}{16}$ is used to trade off increasing the effect of

---

[4]The correspondence problem in the use of multiple images was not solved. Ikeuchi lists this option as a future research direction.

smoothness, in terms of $Z^*_{ij}{}^n$, versus increasing the effect of the accuracy $(I_{ij} - R(F^*_{ij}{}^n, G^*_{ij}{}^n))(\frac{\partial R}{\partial Z})|_{F^*_{ij}{}^n, G^*_{ij}{}^n)}$ where Z is either F or G. Increasing the values of $\frac{\lambda}{16}$ increases the accuracy at the cost of decreased smoothness. It should be noted that Ikeuchi used in his paper [Ikeuchi 80a] a non standard notation of $\frac{\partial R}{F^*_{ij}}$ and $\frac{\partial R}{G^*_{ij}}$ instead of the more standard notation of $\frac{\partial R}{\partial F}$ and $\frac{\partial R}{\partial G}$ that is used below.

$$F_{ij}{}^{n+1} = F^*_{ij}{}^n + \frac{\lambda}{16}(I_{ij} - R(F^*_{ij}{}^n, G^*_{ij}{}^n))(\frac{\partial R}{\partial F^n_{ij}})|_{F^*_{ij}{}^n, G^*_{ij}{}^n)}$$

$$G_{ij}{}^{n+1} = G^*_{ij}{}^n + \frac{\lambda}{16}(I_{ij} - R(F^*_{ij}{}^n, G^*_{ij}{}^n))(\frac{\partial R}{\partial G^n_{ij}})|_{F^*_{ij}{}^n, G^*_{ij}{}^n)}$$

where (2–2)

$$F^*_{ij} = \frac{F^*_{i+1\,j} + F^*_{i-1\,j} + F^*_{i\,j+1} + F^*_{i\,j-1}}{4}$$

is used as an average value of F.

$$G^*_{ij} = \frac{G^*_{i+1\,j} + G^*_{i-1\,j} + G^*_{i\,j+1} + G^*_{i\,j-1}}{4}$$

is used as an average value of G.

The major advantage of this method is that it can generate different orientations for each texel and its surrounding surface patch, i.e. it can create curved surfaces. This is gained at the cost of requiring

- a priori knowledge of the texels' shape
- that orientation constraints that must be recovered from the occluding boundaries inward,
- and at the cost of considerable computation.

) In conditions in which no such knowledge exists, or where the method does not

converge, this method is inapplicable.

This work showed that an iterative constraint propagation method can be used to construct non-planar surfaces from planar surface patches by means of shape from texture information. It also showed that the results of shape from shading research can be extended to the domain of texture.

2.3.2 Uniform Size Assumed to Derive Planar Surfaces

The next approach, due to Ohta [Ohta et. al. 81], uses the weaker assumption that the size of the texels is uniform rather than, as in Ikeuchi's, that the size of the texels are uniform and known. Ohta described how the 2-D Affine transform can be used to approximate the perspective effect and thus obtain a method for recovering orientation.

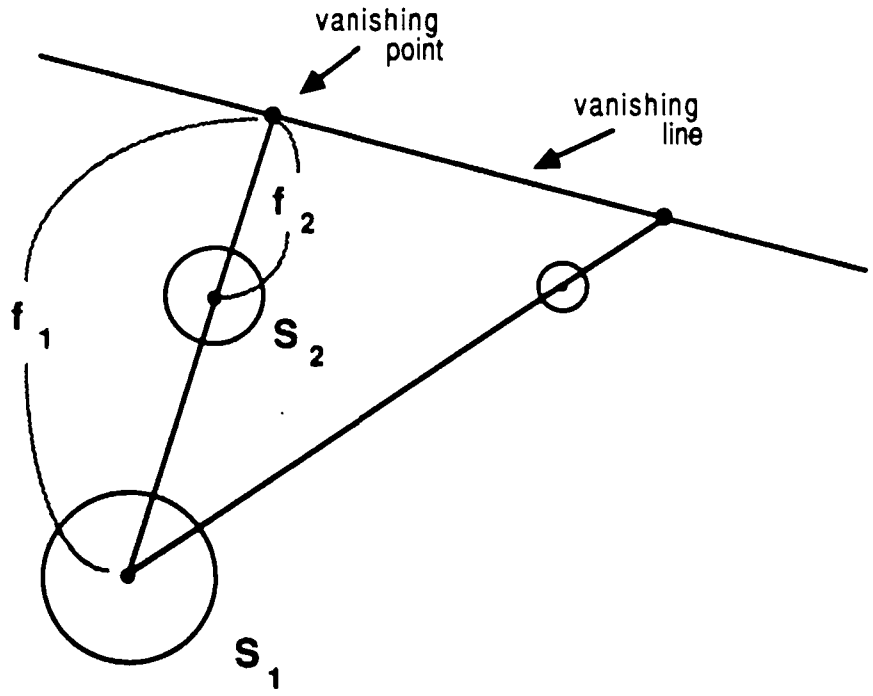


Figure 2-4: How texel size can be used to derive orientation
adapted from [Ohta et. al. 81]

Given two texels T_1 and T_2 with areas of S_1 and S_2 respectively, the ratio of

distances from the center of mass of the texels to a point on the vanishing line (see figure 2-4 for a graphical representation) is defined to be:

$$\frac{F_1}{F_2} = \frac{S_1^{1/3}}{S_2^{1/3}} \tag{2–3}$$

As described earlier (see page 15) a single vanishing point is equivalent to having only one orientation constraint; to derive a unique orientation for a surface patch at least one more texel would have to be used. Thus if the texels can be separated (an assumption similar to that of Kender's shape-from-uniform-texel-spacing cue), their sizes can be calculated, and the orientation of the surface is simple to calculate by this algorithm. This approach, given the constraints of equal sized texels and the ability to separate the individual texels, has been tested on real images (see chapter 6).

2.3.3 Uniform Texel Size for Non-Planar Surfaces

Aloimonos [Aloimonos and Swain 85] fuses the two previous approaches into a single more robust method. He generates an approximation to the orientation of locally planar surfaces by Ohta's method and then generates a potentially non-planar surface by an iterative propagation algorithm, similar to that of Ikeuchi's (see page 18). The method trades off overall smoothness of the surface against accuracy in local orientation as originally measured. The estimated p and q for some surface patch whose image point at (i,j) on the $(n+1)^{th}$ iteration of Aloimonos's algorithm is defined to be:

$$p_{i,j}^{n+1} = pa_{i,j}^{n} + \omega \, [\, I_{i,j} - R(pa_{i,j}^{n}, qa_{i,j}^{n}) \,] \, \partial R/\partial p$$

$$q_{i,j}^{n+1} = qa_{i,j}^{n} + \omega \, [\, I_{i,j} - R(pa_{i,j}^{n}, qa_{i,j}^{n}) \,] \, \partial R/\partial q \tag{2–4}$$

where $pa_{i,j}$ and $qa_{i,j}$ are the average values of p and q around the point (i,j)

(equivalent to Ikeuchi's $F^{*}_{i,j}$ and $G^{*}_{i,j}$); R is a distortion map which Aloimonos calls the textural "reflectance" (as in Ikeuchi's method), and ω (similar to Ikeuchi's $\frac{\lambda}{16}$) gives the error in the textural accuracy map relative to the smoothness.

Aloimonos's method approximates the non-planar surface by iterative constraint propagation. It uses constraint propagation not as a means of determining a second constraint on the orientation of the surface patches (as in Ikeuchi's method see page 18) but rather to generate a smooth surface from the planar patches. The method has the advantage of deriving the shape of non-planar surfaces without the major disadvantage of requiring a priori knowledge of the shape of the texels. Unfortunately, this iterative approach is not always convergent. Provably convergent iterative algorithms for solving systems of equations similar to that of Aloimonos are known to be difficult to devise [Lee 87]. The lack of assured convergence limits the applicability of any such relaxation method.

Furthermore, due to the effects of the heuristically set weighting factor ω, the method displayed the following undesirable behaviors on simple synthetic images according to Aloimonos:

1. The local depth of convex objects tended to be overestimated while the depth of concave objects were underestimated. However as the surface becomes most nearly perpendicular to the image plane these estimation errors are minimized.
2. The orientation of boundary texels were not allowed to change during the iteration phase. Errors due to calculating the orientation of boundary texels were the predominant factor in influencing the total error of the method.

The requirement of a heuristically set weighting factor, plus the behavior problems described above, limit the applicability of this approach.

2.3.4 Uniform Texel Size for Natural Textures

Blostein [Blostein 87; Blostein and Ahuja 87] derived a method of calculating the orientation of naturally textured surfaces based on the assumption that the texels are of approximately uniform size. "Natural textures" are defined as containing texels [Blostein and Ahuja 87, page 444] that are either

1. partially occluded or
2. are themselves textured at finer scale.

These two aspects of the texture make the separation and identification of individual texels difficult. To solve this problem texel extraction is integrated with orientation calculation.

Texels are extracted by a multi-scale region detector that models candidate texels as disks of uniform grey-scale. The image I is convolved with two variations on the Laplacian of the unnormalized Gaussian filter, $\nabla^2 G*I$ and $\frac{\partial}{\partial\sigma}\nabla^2 G*I$, at different filter sizes of σ. Since both filters respond proportionally to lighting contrast changes, dividing one value by the other results in a contrast-independent measure. Disks are postulated to lie at a pixel if their "diameter" is calculated to be:

$$D=2\sigma\sqrt{\sigma(\frac{\partial}{\partial\sigma}\nabla^2 G*I)/(\nabla^2 G*I)+2}$$

where:

$$2\sqrt{2}\sigma - 2 \le D \le 2\sqrt{2}\sigma+2$$

Candidate texels are constructed by generating all subsets of disks and selecting those disks that fulfill spatial connectivity criteria. The contrast of each candidate texel is utilized in determining how large a contribution each candidate texel can provide in calculating the best fit planar surface. The region contrast of each candidate texel is determined from its constituent disk contrast C where:

$$C = \frac{2\sigma^2}{\pi D^2} e^{D^2/8\sigma^2} (\nabla^2 G * I)$$

Blostein's region contrast measure *C* can be considered to be a precursor to the intra-cue correctness factor that will be described in section 3.2.6. The contrast measure would seem less effective than the intra-cue correctness measure since it is based on a factor which is unrelated to the method used to generate the orientation constraint.

The planar fit of a surface to these candidate texels is created by a coarse-to-fine region fitting method. For each potential planar fit, the area of each candidate texel is compared with the texel area predicted by the parameters (A_c,S,T) where S is the slant, T is the tilt, and A_c denotes the area of a texel located precisely at the image center and chosen from a set of preset values. For a coarse fit the set is {10,20,40,80,160,320,640} and for a fine fit A_c is chosen from a set of values which increase incremental by less than 25%.

The expected area for a particular choice of (A_c,S,T) is calculated as:

$$A_i = A_c(1 - tan\,(\frac{x_i}{focal\ length})\ tan\ S)^3 \qquad (2\text{–}5)$$

for an image point with coordinates (x_i, y_i, focal length). The support for each potential planar surface orientation is constructed by adding contributions from each potential texel. The contribution of each region to the fit-rating is based on the region area, how closely the region's area fits the predicted size, and on the region contrast:

$$fit\text{–}rating = \sum_{regions} A\ c\ e^{-r^2/4} \qquad (2\text{–}6)$$

where:

$$c = |\ region\ contrast\ |$$

$$A = (region\ area)$$

$$r = \frac{max(expected\ area\ ,\ actual\ area)}{min(expected\ area\ ,\ actual\ area)}$$

The region contrast is included on the premise that regions of higher contrast are more important perceptually, even though experimental results (see [Blostein 87] page 41) seem to point to the contrast being unnecessary. The fit-rating with the largest value is chosen as the estimate of the textured surface orientation.

This approach derives the orientation of the surface, while discriminating texels from other aspects of the texture and the image. In order to perform these two operations together the following limitations are imposed:

1. The method is only applicable to planar surfaces. As the surface diverges from planarity the resulting best fit degrades.
2. The image must be segmented into surfaces for this shape-from-texture method to be applicable.
3. The textural elements are approximated by disks. As the texture elements become more structured, this approximation becomes more erroneous and the best fit planar surface diverges from the correct orientation.

2.4 Texel Isotropy

The shape of a textured surface can also be derived from the effects of foreshortening distortion on the textural primitives. If the density of the texels is sufficiently large, then this distortion can be approximated from measurements on texel edges. Two major assumptions need to be made about the texture:

1. The major component of the distortion in the texture is due to foreshortening compression.
2. The texture is considered to be a distribution of edge directions and those edge directions are assumed to be uniform prior to the effects of projection. This is important since this limits the types of textures to which this approach is applicable.

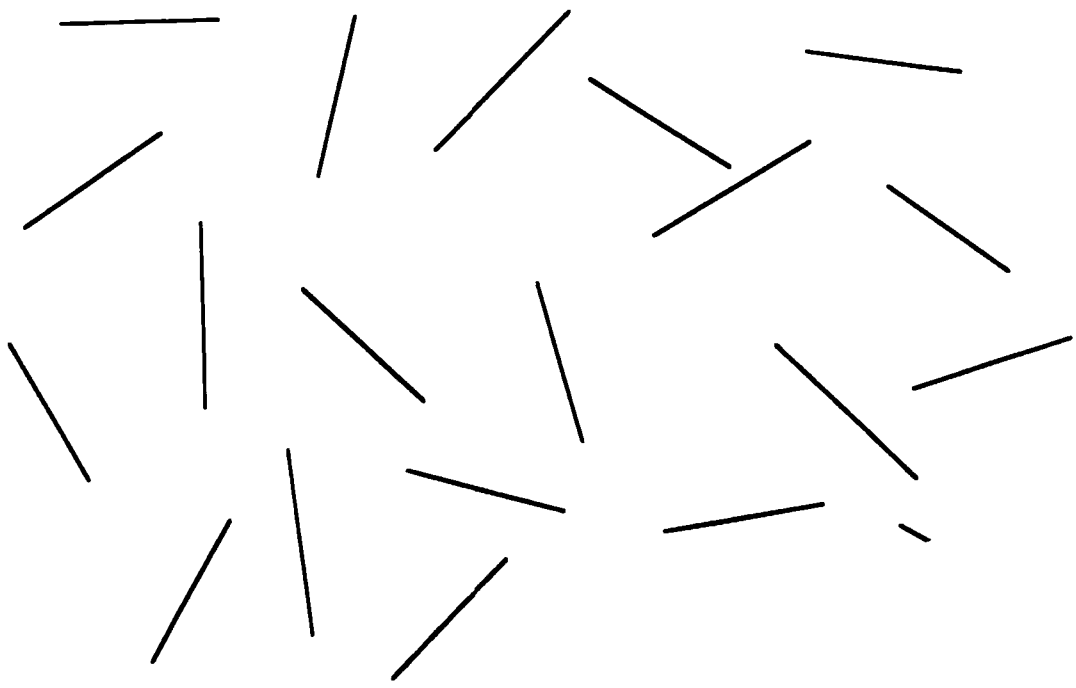


Figure 2-5: Isotropic Texel edges

2.4.1 Texel-Isotropy Based on Statistical Approximations

Witkin [Witkin 80] derived a method, based on a statistical approximation, that attempts to distinguish the distortion effects of perspective from the intrinsic properties of the texture. Witkin's method (as corrected and improved by Davis et.al. [Davis, Janos and Dunn 83]) is based on the additional assumptions that the foreshortening compression can be modeled by orthographic projection. The orthographic projection of the original pattern consists of scaling the pattern by cos σ in the direction of τ where σ and τ are the angles of slant and tilt respectively. This assumption of orthographic projection is most valid when the surface is "far away" and "near" the center of the image.

The algorithm consists of two stages. The first stage determines the edges in the image and their associated edge directions; the second derives the orientation of the surface patches from the edges. During the first stage the edges are recovered from the zero-crossings contours of a $\nabla^2 G$ convolution.

However, zero-crossings do not necessarily uniquely define the direction of the edge at any point; one may have to choose among several zero-crossings [Marr and Hildreth 79] [Brown 86]. The two major solutions to this consist of either choosing the zero-crossing with the maximum slope, or the zero-crossing whose orientation agrees with the orientation of other neighboring zero-crossings[5].

The next step consist of finding the most likely orientation. A simplified approach due to Davis et.al. [Davis, Janos and Dunn 83] uses a histogram of the edge directions. In this approach the likelihood $L(\sigma,\tau;A)$ for any set of sample counts, $A = \{\alpha_1,\alpha_2,...,\alpha_n\}$, of the projected directions is defined as:

$$L(\sigma,\tau:A) = \frac{sin\ \sigma\ cos^n\sigma}{\prod_{i=1}^{n} [1 - sin^2\ \sigma\ sin^2\ (\alpha_i - \tau)]} \qquad (2\text{–}7)$$

The slant and tilt with the largest value as generated above is the most likely orientation.

The method has shown success on a small group of both synthetic and real images. The method is limited as follows:

1. In general the accuracy of the orientation estimate is proportional to the amount of slant in the surface. The method is better able to recover the slant than the tilt. Moreover, it tends to overestimate both the slant and tilt, and their consistency depends on the homogeneity of the texture [Dunn 84].
2. Discontinuities in depth and multiple surfaces can not be recovered directly by the method. One solution that has be given [Dunn 84] is to divide the image into areas; the orientation of each of the areas can be separately calculated. This approach has the same problems as those described in the shape-from-uniform-density section (see page 33).

The accuracy of the method is further limited by the accuracy to which the edge directions can be calculated [Dunn 84] and how well the edges can be

---

[5]According to the theorem by Marr and Hildreth [Marr and Hildreth 79] called *the condition of linear variation* in smooth images, the two strategies are equivalent.

distinguished, just as texels need to be distinguished in the shape-from-uniform-space methods.

2.4.2 Texel Isotropy by Parallel Scanning Lines

Another method for deriving the orientation of surfaces based on the assumption of texture isotropy is due to Kanatani [Kanatani and Chou 86; Dunn 86]. In this method the distribution of tangent directions is related to a probability distribution. Thus orientation can be calculated without having to actually determine the tangent directions of individual texture edges.

A number of parallel "scanning" lines at different orientations are superimpsed on the image. The number of intersections of texture edges with the scanning lines are counted and divided by the length of the lines to get the intersections per unit length. $\mathbf{N}_k$ is defined as the number of intersections per unit length as the orientation $\frac{\pi k}{\mathbf{N}}$ for $0 \le k < \mathbf{N}$. The slant and tilt are then computed:

$$\begin{aligned} \sigma &= \pm 2((A_2)^2+(B_2)^2)^{\frac{1}{4}} \\ \tau &= \tfrac{1}{2}\, tan^{-1}\tfrac{B_2}{A_2} \quad if\, A_2<0 \\ \tau &= \tfrac{\pi}{2} + \tfrac{1}{2}\, tan^{-1}\tfrac{B_2}{A_2} \quad if\, A_2>0 \end{aligned} \tag{2–8}$$

where A_2 and B_2 are the first two coefficients of the Fourier series:

$$A_2 = \frac{\sum_{k=0}^{N-1} N_k \cos\left(\frac{2\pi k}{N}\right)}{\sum_{k=0}^{N-1} N_k}$$

$$B_2 = \frac{\sum_{k=0}^{N-1} N_k \sin\left(\frac{2\pi k}{N}\right)}{\sum_{k=0}^{N-1} N_k}$$

2.4.3 Texel Isotropy from Autocorrelation

The next method, due to Brown and Shvaytser [Brown and Shvaytser 88], also assumes that the texture is isotropic but recovers surface orientation from the autocorrelation of the textured image. As in Witkin's cue, this cue is global and structural in nature. The difference lies in the fact that Witkin's method used the histogram of edges while this method uses the autocorrelation function.

Given the assumption that the texture is isotropic the autocorrelation of a texture is "foreshortened" in the same way as the texture itself. The image surface's Gradient is given by a simple operation applied to the "image" of the autocorrelation function at a point.

Autocorrelation-based shape-from-texture can be considered half-way between shape-from-texel-isotropy and shape-from-density. This is because it measures the isotropic aspect of texture foreshortening by a density-like measure. This "cross-over" aspect of Brown and Shvaytser's cue is an indication that the diverse range of shape-from-texture cues can be consolidated. This consolidation, which is discussed in greater depth in section 2.7, separates the cue-specific algorithm from the texture measure itself.

Brown and Shvaytser's autocorrelation approach has two advantages over the previous approaches:

1. The previous two methods first applied an edge operator to the image and then measured the orientation in terms of the foreshortening distortion on the edges. The quality of those results is thus based on the quality of the edge operator.
2. The autocorrelation method uses the whole range of intensities in the image rather than just sharp edges. Since only a fraction of the pixels in an image are edges, edge-based methods are less resilient to the effects of noise.

The surface orientation is calculated in two phases. During the first phase the autocorrelation in a region of the image is calculated. Given a grey scale image F, where the vector $R = (x,y)$ denotes a point in the image plane, and the

grey scale value of the point is defined as F(R), the autocorrelation A(R), is defined as:

$$A(R) = \int (F(r') - M)(F(R' + R) - M)dR' \qquad (2\text{–}9)$$

where M is the mean value of the image region and R = (r1,r2). The surface orientation is calculated from the autocorrelation moment matrix

$$u = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix}$$

where :

$$u_{ij} = \int (r_i r_j A(R))\, dR.$$

Under the simplifying assumption that orthographic projection is valid (equivalently, that the distance between the surface plane and the observer is very large in comparison to the linear size of the portion of the surface being worked on) the slant σ and tilt τ are derived:

$$\sigma = arccos \sqrt{(u_{11} + u_{22} - ut)\,/(u_{11} + u_{22} + ut)}$$

$$\tau = \tfrac{1}{2}\, arctan 2 \frac{u_{12}}{u_{11} - u_{22}}$$

where :

$$ut = \sqrt{(u_{11} - u_{22})^2 + 4u_{12}}$$

2.4.4 Texel Isotropy - Summary

In this section three different shape-from-texel-isotropy cues have been described. Each of the cues utilizes a different algorithm for measuring texel isotropy and therefore generates results with different accuracy depending on the

image conditions. In general, each of the cues require the existence of an image segmentation phase to guarantee locally planar segments; this limits the applicability of the cues. Images containing multiple textures or multiple surfaces are problematic.

2.5 Uniform Density

The previous shape-from-texture cues were based on measuring a "structural" change in the texture and thus recovering orientation information. This structural quality of the cues is most apparent in methods such as shape-from-uniform-texel-size. The structural aspect decreases when more than one texel is required such as in the shape-from-texel-isotropy cues, specifically Brown and Shvaytser's autocorrelation-based method. In the next group of cues individual texels may not show perspective distortion; rather, the effects of perspective distortion are required by statistically-based measurements.

The difference in the statistical cues vs. the structural cues is due to differences in the specific textures on which each set of cues operate. If the texels are sufficiently dense that they can not be separated, then both the shape-from-texels-spacing cue and the shape-from-texel-size cue will fail. The next group of methods utilize a different but related assumption, that the change in the density of texel edges can be used to calculate the orientation of surfaces.

2.5.1 Uniform Density Measured by Change in Edge Density

The first method, due to Aloimonos [Aloimonos and Chou 85; Aloimonos 86] assumes that the density of edges is the same everywhere in the unprojected surface and derives a constraint on the value of p and q from the change in the measured edge density between areas of the image. The image is first separated into patches. In each patch the edge density is calculated[6] and then for every

[6]Aloimonos does not describe what method is used to identify and count the edges but rather points the reader to [Marr and Hildreth 79; Marr and Poggio 79; Bandopadhyay 84].

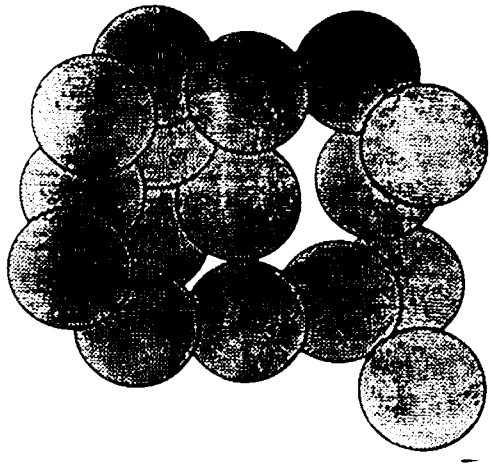


Figure 2-6: A textured surface in which texels overlap

group of two patches in the image a constraint on the value of p, and q is defined to be :

$$\frac{1 - A_1 p - B_1 q}{1 - A_2 p - B_2 q} = \left(\frac{K_2 S_1}{K_1 S_2}\right)^{\frac{2}{3}} \tag{2-10}$$

where $(A_n, B_n, -1)$ is defined to be the center of the image region n; K_n is defined to be the total count of edges in image region n and S_n is the area of image region n.

The image is arbitrarily disected into patches, the size and placement of which are heuristically defined. This creates three problems:

1. Edges can cross the patches, thus making the density an approximation. This is partially resolved for planar surfaces by averaging the orientation results determined by all the patches in the image.
2. The size of the image regions should be defined based upon texel size and spacing criteria rather than predefined for all surfaces. This is important because of two conflicting potential errors:
    - If the size of the image regions is small in proportion to the size of the texels then the number of edges per region will be small and small digitization errors can create large

orientation errors.

- If the regions are set too large and the surface has a sharp orientation, then density changes occur within the image region themselves.

3. The patches are arbitrarily assigned to the image and therefore do not take into account changes in texture or orientation, and they can cross surface boundaries.

If the method is applied to a group of textured images some of which are small and densely packed while other textures are large and loosely packed one finds that the error statistics of the method increase as the texture size and spacing increase.

2.5.2 Shape-from-Texel-Isotropy into Shape-from-Uniform-Density

Dunn [Dunn 86] creates a shape-from-uniform-density method by modifying Kanatani's shape-from-texel-isotropy algorithm. He replaces the texel isotropy assumption by the assumption that the "statistics of the spatial probability distribution are stationary" [Dunn 86 page 47] - the assumption of uniform texel density. Furthermore, by assuming that the surface is a *developable surface* (that is, one of the principal curvatures is zero), simple curved and non-planar surfaces are solvable.

The basic algorithm consists of segmenting the image into multiple overlapping bands. The bands are placed at different positions and have varying orientations (as in Kanatani's method, page 28). Within each band the expected density is estimated by the average arc length (or texel edges) $\mathbf{E}(\rho,\phi)$ per unit length of the center line of the band. This method differs from that of Aloimonos's method (see page 21) in the formulation of the areas as well as in the specific method for determining the density: edges crossings in bands for Dunn, vs. edge finding and density calculation for Aloimonos.

The surface orientation is recovered by first determining the direction orthogonal to the tilt. If the magnitudes of density in opposite directions are

constant, then the tilt direction has been found. The cosine of the slant direction is calculable from the average arc length and one of the Gram polynomials[7].

This method has the advantage over the previous shape-from-uniform-texel-density method (page 33) of deriving surface orientation for singly curved surfaces. It is still limited in the same ways as the previous method:

1. Edges can cross the patches thus making the density measure approximate.
2. The size of the bands are preset rather than a function of the density and therefore they approximate the true curvature of the surface.

2.5.3 Uniform Density measured by the Wigner Distribution

The third density-based shape-from-texture method, due to Jau and Chin [Jau and Chin 88] uses the Wigner distribution (or transform) to compute the texture gradient of a surface. The Wigner distribution of a two dimensional image is a four dimensional function that represents an image in both space and spatial-frequency. Theoretically, this supplies the instantaneous texture gradient for each and every pixel in the image. However, the spatial-frequency is calculated, for each pixel, within a window surrounding the pixel.

Jau defines a pseudo Wigner Distribution (or WD) for discrete images in which the spatial-frequency is calculated for a square window surrounding each point (m,n) with size S x S, defined by $b(k,l) = 0$ if $|k| \geq S'$ or $|l| \geq S'$ where $S = 2S' - 1$. The WD is defined by:

---

[7]The specific formulation of the slant and tilt are left out of this survey since they require numerous definitions beyond the space available. For further information on these formulas see [Dunn 86].

$$W_f(m,n,u\frac{\pi}{2},v\frac{\pi}{2}) =$$

$$4 \sum_{k=S'+1}^{S'-1} \sum_{l=S'+1}^{S'-1} e^{-j2\pi(ku+kv)/S} f(m+k,n+l) f^*(m-k,n-l)b(k,l)b^*(-k,-l)$$

where f(m,n) is the image function at the image point (m,n); u and v are discrete variables in the frequency domain. f* is the complex conjugate of f.

The texture density at the point (m,n) is defined as

$$d(m,n)=1-\frac{E_l(m,n)}{E(m,n)} \quad 0 \leq m, n \leq N-1$$

where the total energy at the point is defined as

$$E(m,n) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} W_f(m,n,u,v)$$

and the low frequency energy content is determined by:

$$E_l(m,n) = \sum_{(u,v) \in \Omega} W_f(m,n,u,v)$$

Ω is the low frequency band in the spatial-frequency domain given by

$$\Omega=\{ (u,v)|0<2 \pi \sqrt{u^2 + v^2}<w\}$$

and *w* is a pre-determined heuristically set threshold value.

This shape-from-uniform-texel-density cue is able to compute orientation information even when the texels are inseparable allowing the method to be applied to a wider range of imagery. Unfortunately this advantage is lessened by a number of disadvantages of the cue:

1. The cue contains a number of variables whose value is not prescribed by the theory but is rather determined experimentally.
2. The authors state that the cue does not require the selection of a window size, yet inherent in the final algorithm is a heuristically set window size.
3. The Pseudo Wigner Distribution is a discrete approximation and has not been tested on real imagery.
4. The cue is very computationally expensive. (In fact, Jau and Chin discuss how the method could be sped up using optical processors).
5. The method is only applicable to pre-segmented imagery of planar surfaces.

2.5.4 Uniform Density - Summary

In this section three shape-from-uniform-texel-density cues have been described. Each of the cues measures a change in the number of texture elements per unit area. This density assumption can be considered as a variation on the uniform texel size assumption under conditions in which the texels need not be separable. In fact, they need not be of uniform size. Effectively this trades off increased generalization against a larger sampling area.

Each of the cues measures the texel density in a different manner: number of edges per area, number of edge intersection with scan lines, and a Wigner distribution measure. Since the density is measured in a different way, different image characteristics will generate better or worse results for each of the three cues.

2.6 Use of Multiple Assumptions

Previous researchers have created methods that "fused" more than one shape-from-texture cue. These methods fused only the requirements of the cues rather than fusion of the cues themselves. Effectively these were systems that are more limited in their domain of applicability than the individual cues.

Two different shape-from-texture methods have been created by utilizing two assumptions about the texture: Bajcsy [Bajcsy and Lieberman 76] utilizes the assumption that the texels are uniformly spaced and uniformly sized, while Pentland [Pentland 86] assumes texel isotropy and uniform texel spacing. In both of these methods the image must fulfill both of the assumptions in order that the shape-from-texture method correctly derives orientation. Thus each method has a more limited domain than any method based on only a single assumption.

2.6.1 Assuming Texel Isotropy and Uniform Texel Spacing

As described above (see page 27) texel isotropy is better able to derive tilt than slant. Pentland [Pentland 86] devised a method utilizing a combination of texel isotropy[8] (as in Witkin's method page 26) and shape-from-texel-spacing to derive surface orientation. In this method the tilt component of surface orientation is recovered directly by Witkin's approach while the slant component is generated by first creating a regional estimate of the slant from a shape-from-texel-spacing method (as in Kender's method page 15). This estimate is then compared to a texture measure along the apparent maximum foreshortened direction to generate a more exact slant measure.

This approach is in many cases no better than that of Witkin's since it requires not only texel isotropy but also textural homogeneity over regions of the image, plus additional constraints due to the shape-from-uniform-texel-spacing methods. These assumptions highly limit the types of textured surfaces to which this method is applicable.

---

[8]Pentland also described a tilt calculation method similar to his shape from shading method that is very highly restricted. It makes the assumptions that the texture is homogeneous, non-planar, has a constant albedo, and constant illumination. These restrictions are even too strong for Pentland, who offers the use of texel isotropy as an alternative.

2.6.2 Assuming Uniform Texel Spacing and Uniform Texel Size

Bajcsy [Bajcsy and Lieberman 76] formulated and implemented the first shape-from-texture method utilizing fourier descriptors of texture. The fourier transform measures major variations in the rough shape class, size, and spatial organization of texels. By dividing the image into equally sized windows, changes in spatial frequency can be measured across windows and thus the surface gradient can be determined.

Since the gradient calculation is based on differences of fourier features between windows, the size of the window is of major import. As discussed earlier, the size and placement of windows should be defined differently for each image based upon characteristics of the surface orientation and texture.

2.6.3 The Multiple Shape-From-Texture Approach

The final shape-from-texture method, which we present in this work, is based on the fusion of multiple shape-from-texture cues into a single robust system [Moerdler and Kender 87a; Moerdler and Kender 87b]). The process of fusing orientation constraints and generating surfaces can be broken down into the following three phases:

1. The creation of surface patches containing one or more texels or "texel patches" and the generation of multiple orientation constraints for each patch.
2. The unification of the orientation constraints per patch into a "most likely" orientation.
3. The formation of surfaces from the texel patches.

The first phase of the system consists of several shape-from-texture components which generate augmented texels. Each augmented texel consists of a texel patch, orientation constraints for the texel patch, and a correctness weighting per constraint. The correctness weighting is defined separately for each shape-from method and is comprised of two components: an intra-cue correctness factor, and an inter-cue normalization factor.

A separate intra-cue correctness factor is generated for each orientation constraint in each augmented texel. It is based on a heuristic measure of how well the texels that were used in calculating the constraint have fulfilled the underlying assumptions of the shape-from-texture cue that generated the constraint.

The inter-cue normalization factor is generated across constraints, but still within each augmented texel. It assures that the sheer number of constraints generated by each cue does not overly affect the systems choice of what is the correct orientation.

Once the orientation constraints have been generated for each augmented texel, the next step consists of unifying the constraints into one orientation per augmented texel. The major difficulty in deriving this "most likely" orientation is that the constraints are usually errorful and inconsistent. A simple and computationally feasible solution to this is to use a Gaussian sphere as a Hough accumulator array for constraint fusion. The Gaussian sphere defines all possible visible and invisible surface orientations. Each orientation constraint circumscribes a great circle on the Gaussian sphere; two different constraints generate two great circles that intersect at two points uniquely defining the orientation of both the visible and invisible sides[9] of the surface patch. By applying all of the weighted constraints to the sphere, the point with the highest weighting can be calculated and thus the solution simply determined.

The Gaussian sphere is approximated by a hierarchically tesselated Gaussian sphere based on triangular-shaped faces called trixels (See figure 4-3) [Fekete and Davis 84; Korn and Dyer 86]. The top level of the hierarchy is the twenty equilateral triangular-faced icosahedron. At each successive level of

[9]This mapping of potential surface orientations onto the Gaussian sphere and the related generation of both visible and invisible orientations are related to such psychological phenomenon of Necker reversal.

refinement each trixel has four triangularly shaped children, which more closely approximate the surface of the spherical surface than their parent. This hierarchical methodology allows the user to specify the accuracy to which the orientation can be calculated by defining the number of levels of tesselation that are created. The system generates the "most likely" orientation for each texel patch by accumulating the evidence from all the orientation constraints (generated in phase one) for the patch in this triangular-segmented sphere of orientations.

This method does not assure that under all circumstances a single "most likely" orientation will be derived. When more than one "most likely" orientation is derived for a patch, the system performs a Waltz-type filtering. It computes the "most likely" orientation for other neighboring augmented texels, and then re-analyzes the orientation constraints for those texels that do not have a single "most likely" orientation. The system reanalyzes the unsolved texel patches by a three step constraint pruning process:

1. It considers all of the patch's constraints, and computes which other patches are used to generate each constraint. These are considered related patches.
2. If a constraint's related patch has been solved, but one of its own constraints does not correctly define the "most likely" orientation of the related patches then that constraint is "removed". A constraint is effectively removed by setting the constraint's weighting factor to zero.
3. Once this constraint pruning has occurred, the system recomputes the "most likely" orientation for the patch.

This re-analysis does not assure a single "most likely" orientation either, but it does aid in simplifying and deriving a single "most likely" orientation for a large number of surface patches.

This approach solves a wider range of images containing textured surfaces than any of the previous shape-from-texture methods. The specific details and reasons for each stage of this approach will be described in the remaining

chapters, with examples of the system operating on real and synthetic images displayed in chapter 5.

2.7 The Inter-Relationship Between Cues

Each of the shape-from-texture cues calculates orientation constraints based on a measured change in some aspect of the texture between regions of the image. In doing so each of the cues is effectively defining a segmentation of the image into "measurement-units".

> **Definition 2-1:** The measurement-unit is the minimum sized region used by the shape-from-texture cue to measure a change in a texture property.

For example, in shape-from-uniform-texel-size the measurement-unit is a segmented texture element, while in shape-from-uniform-texel-density the measurement-unit is a window. The three major measurement-units of blobs, edges, and windows measure the basic image primitives of blobs, edges and points.

The fact that these three types of measurement-units repeat throughout the cues is important and allows the following observation. Given any of the cues (for example shape-from-uniform-texel-size), a new cue can be created by replacing the existing measurement-unit with a new measurement-unit. Effectively this modification allows an existing cue to be applicable to a different range of textures. This will be exploited in the multiple shape-from-texture fusion paradigm (see section 2.7).

Previous shape-from-texture classification methods have been based on separating cues into two categories, structural (e.g. Kender's method, Ikeuchi's method, Ohta's method, Aloimonos's method etc.) and statistical (e.g. Brown's method, Jau's method, etc.) Structural methods measure changes in the structure of a specific texel while statistical methods use measured changes

Cues	Page Number	Masurement Area	Method of Extraction
Shape-From-Uniform-Texel-Spacing			
Kender's method	page 16	Blobs	Blob finding
Shape-From-Uniform-Texel-Size			
Ikeuchi's method	page 17	Blobs	A priori known Blobs
Ohta's method	page 20	Blobs	Blob finding
Aloimonos' method	page 21	Blobs	Blob finding
Blostein's method	page 23	Blobs	Approximated by disks of uniform grey-scale
Shape-From-Texel-Isotropy			
Witkin's method	page 26	Edges	Zero-crossing contours
Kanatani's method	page 28	Edges	Parallel scanning lines
Brown's method	page 29	Texture in a window	Autocorrelation of the texture
Shape From Uniform Density			
Aloimonos' method	page 32	Edges	Edge finding
Dunn's density method	page 33	Edges	Arc length of edges intersecting scan lines
Jau's method	page 34	Texture in a window	Wigner distribution to measure density
Use of Multiple Assumptions			
Pentland's method	page 37	Blobs and Edges	
Bajcsy's method	page 38	Texture in a window	

Figure 2-7: Shape-from-texture cues

across groups of texels. With the measurement-unit definition the structural/statistical boundary can be crossed (i.e. a blob-based measurement-unit can be modified to be a window-based measurement-unit).

The existing shape-from-texture cues can now be reorganized as a two dimensional taxonomy of underlying assumption vs. measurement-unit (see figure 2-8) .

The decoupling of the measurement-unit and the underlying assumption simplifies the definition of new shape-from-texture cues. In fact the existing

	Measurement Unit:		
Assumption	**Blobs**	**Edges**	**Windows**
Spacing	Kender's method		
Size	Ikeuchi's method, Aloimonos' method	Aloimonos' Density Dunn's method	Jau's method
Isotropy	Shape-from-eccentricity *	Witkin's method Kanatani's method	Brown's method
Lies on parallel lines	shape-from-parallel-virtual-lines		

* This method is new, created by Kender and Moerdler.

Figure 2-8: A taxonomy of shape-from-texture cues

blanks spaces in table 2-8 point toward future research directions and future cues. For example, texel isotropy has until now been used only for window and edge-based cues. However, given an understanding of the taxonomy, we were able to create a new texel isotropy method that is valid for blobs (see section 3.3.4).

This formalization also simplifies the fusion of multiple shape-from-texture cues. By unifying groups of cues based on their measurement unit, one can utilize a consistent textural element, thus creating a consistent level of abstraction that will allow these different cues to be fused. This will be further discussed in the next chapter.

We can now define a method for generating new shape-from-texture cues based on separating the algorithm from the measurement-unit. Specifically,

> **Definition 2-2:** A new shape-from-texture cue can be created by combining any of the three general types of measurement-units: texel, edges, and windows, and any of the shape-from algorithms.

2.8 Summary

This chapter discussed previous research into the use of texture to derive the shape of surfaces. Many different shape-from-texture methods were described, and classified based on the assumptions that were made about the textured surface in formulating the method. These assumptions limit the class of textures that the methods are applicable to. In fact, the limitations are sufficiently strong that no single shape-from-texture cue is applicable to the complete range of surface textures. A new classification method was created, based on de-coupling the measurement-unit from the general underlying assumption and its specific orientation generation cue. This de-coupling not only defines how new shape-from-texture cues can be created but will simplify the fusion of these cues.

In the next two chapters, a paradigm is proposed for the integration of multiple shape-from-texture cues into a single system. To prove the viability of this approach existing shape-from-texture cues are reformulated based on the paradigm, and the generality of the approach is shown using actual imagery in Chapter 5.

3. Generating Orientation Constraints

This chapter discusses how orientation constraints could be fused; describes a new data structure, the *augmented texel*, that will simplify constraint fusion; and lastly redefines two of the existing shape-from-texture cues, shape-from-uniform-texel-size and shape-from-uniform-texel-spacing, to derive orientation constraints in terms of the *augmented texel* approach. Three additional shape-from-texture cues are also defined herein based on the *augmented texel* approach, shape-from-texel-parallel-lines, shape-from-relative-eccentricity and shape-from-eccentricity.

3.1 Where to Integrate the Information

Knowledge fusion is a diverse field, therefore not all forms of knowledge fusion can be discussed here. For information on the range of different knowledge fusion approaches in computer vision see [Spatial Reasoning and Multi-Sensor Fusion 87]. The more specific field of fusing multiple, overlapping, but potentially conflicting image, information will be discussed, however.

The three basic levels at which shape-from knowledge can be fused are:

1. At the *discrete data element level (pixel-level).* Constraints are generated for each pixel, and after knowledge fusion the resulting (unified) orientations for each pixel must be merged into surfaces.
2. At *the surface level.* Constraints are generated for each surface or surfaces, and then unified into a single orientation for the surface(s).
3. At the *surface patch level.* Constraints are generated for small surface patches (e.g. texel), are unified for each surface patch, and the surface patches are merged into surfaces.

In each of the next three subsections the different fusion levels will be discussed, listing instances in which each approach has been applied, and describing the potential limitations of fusing orientation constraints at that level.

3.1.1 Pixel-Level Knowledge Fusion

In the pixel-level fusion approach, each of the shape-from cues generates orientation constraints for each and every pixel in the image. In practice, this would consist of each cue generating orientation constraints based on measured changes between nearby groups of pixels. The constraints are stored for each of the pixels that comprise the groups of pixels. For each pixel, the system creates a most probable instantaneous orientation for the pixel based on all of the pixels orientation constraints. The individual pixels can then be combined into surfaces (See figure 3-1).

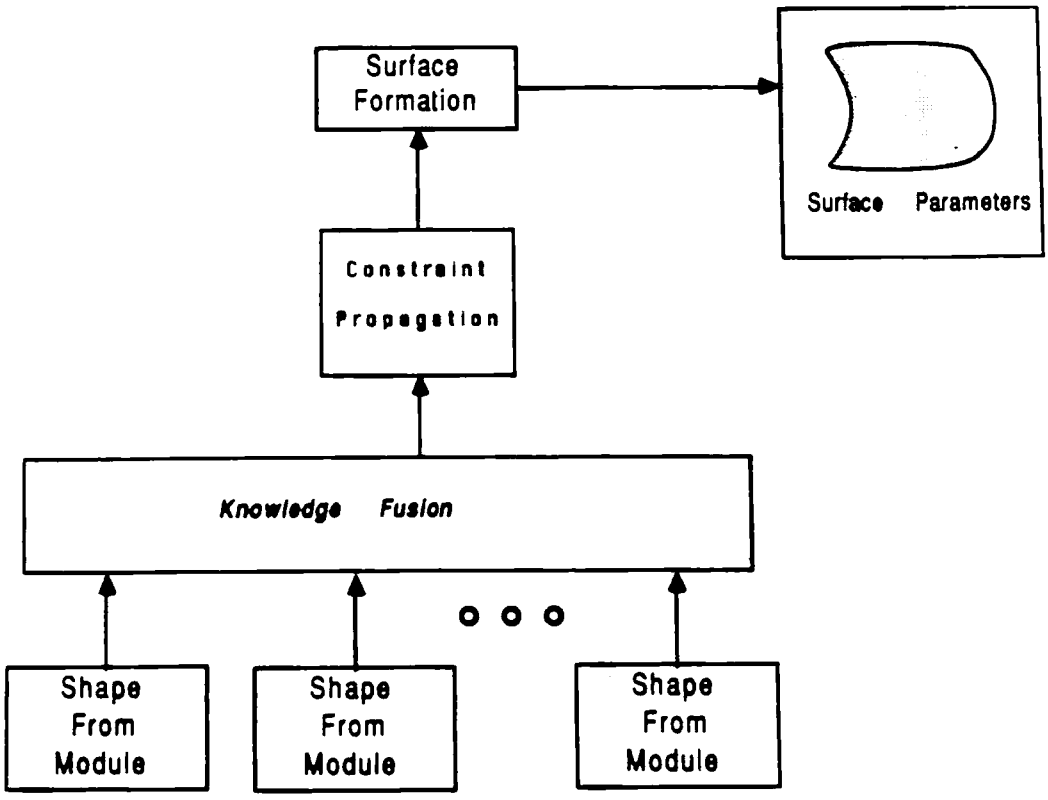


Figure 3-1: Pixel-level constraint generation followed by fusion into surfaces

Since constraints generated for a group of pixels are stored per pixel the problem of how to fuse constraints generated for different overlapping groups of pixels disappears. Furthermore, it simplifies the solving of images containing transparent overlapping surfaces, and/or non-planar surfaces. In both cases the constraints are consolidated per pixel and the pixels formed into surfaces.

Unfortunately, an enormous expense is incurred in the number of pixels for which constraints need to be generated, stored, and fused. This requirement of substantial storage and processing power limits the applicability of the method. Only on a large parallel processor would such an approach be worth even considering (e.g. the Connection Machine which has 64,000 processors [Hillis 81]).

3.1.2 Surface-Level Knowledge Fusion

The second approach consists of generating orientation constraints for each surface in the image. The constraints combined, the surface parameters resolved, and a single consistent representation of each surfaces in the image created (see figure 3-2). Since the fusion occurs at the surface level the constraints are integrated only once for each surface rather than per pixel, substantially decreasing the storage and processing requirements of the system.

Previous shape-from-texture cues, except Ikeuchi's shape-from-apriori-texel-knowledge (see section 2.3.1) and Aloimonos' shape-from-uniform-texel-size (see section 2.3.3) cues, were designed and implemented to derive shape information at the surface level[10]. These surface-level cues assumed that the surface was planar and that the orientation of the surface was a best-fit approximation to the constraints.

The planarity assumption was necessary for most of these cues because

---

[10]Both Ikeuchi and Aloimonos stored constraints for surface patches and performed relaxation between neighboring patches to solve the surface parameters. Therefore they are more closely related to surface patch fusion then surface fusion.

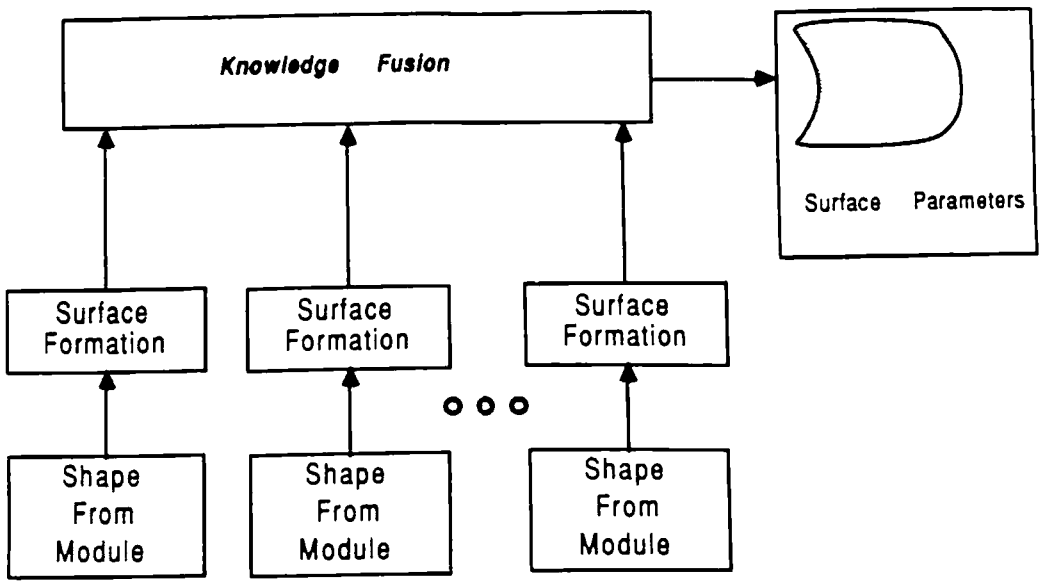


Figure 3-2: Surface parameter generation followed by knowledge fusion

they were statistically based and thus valid only for large samplings of texture or sub-texture elements. Since the constraint data is sparse, it was necessary to utilize the whole image.

In addition to limiting the range of solvable surfaces to those that are planar, this approach has an additional disadvantage. A surface segmentation phase is normally required, and it can not be assisted prior to surface level constraint integration. Unfortunately, surface segmentation is not robust enough in itself. (As will be seen in Chapter 6, robustness can be increased by using texture information.).

Furthermore, the spatial case of overlapping transparent surfaces will be inseparable since pixel level information is lost. Unless a previous texture separation stage exists, this problem will become more difficult if the two surfaces have different textures.

3.1.3 Surface Patch-Level Knowledge Fusion

The last approach, surface patch-level fusion, is midway between the pixel and the surface level. Under this methodology the shape-from-texture methods each generate orientation constraints per surface patch (texel) (see figure 3-3). The constraints are then solved to generate a single orientation per texel and the oriented surface patches are resolved into surfaces.

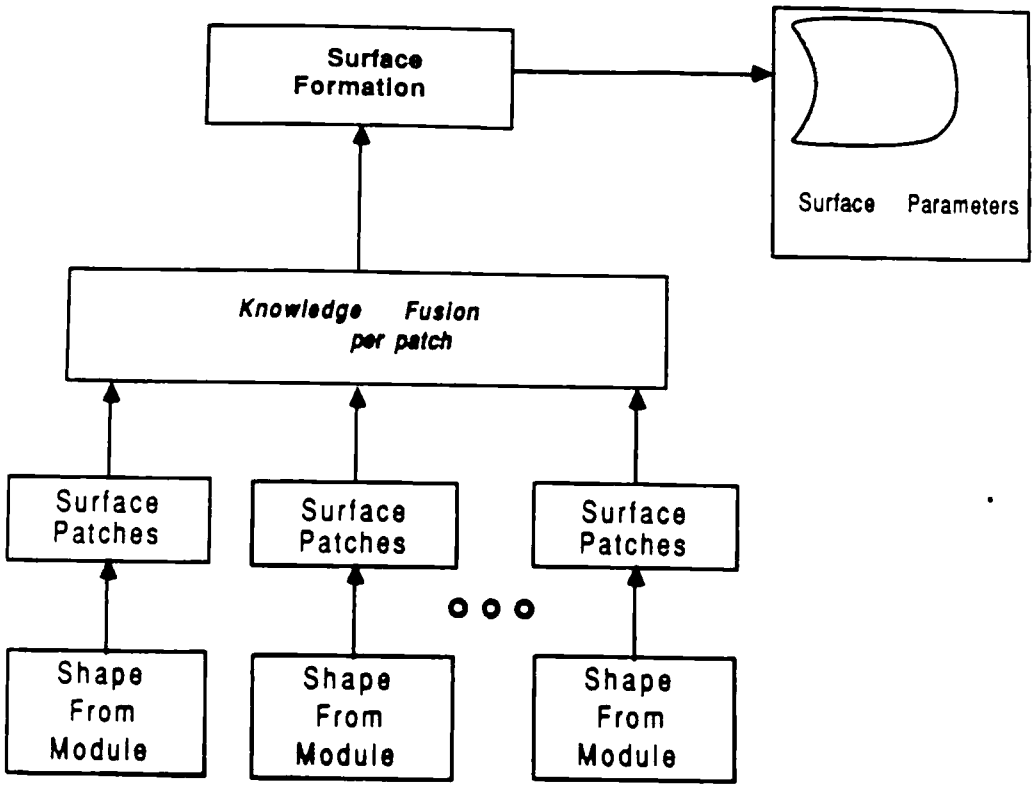


Figure 3-3: Constraints generated per surface patch followed by knowledge fusion and then Surface Reconstruction

This approach limits the data explosion that occurs under the pixel level fusion approach while retaining sufficient surface patch data. It simplifies the

problem of deriving the surface parameters of non-planar surfaces without a priori knowledge. Furthermore, the decoupling of constraint integration from surface generation allows the surface to reflect surface discontinuities that are measured by any of the shape-from methods.

Two previous shape-from-texture systems, those of Ikeuchi's (see section 2-2) and Aloimonos's (see section 2.3.3) generate orientation constraints for surface patches, and then combine the constraints per surface patch using constraint propagation and relaxation to derive a single orientation per surface patch. These approaches were limited to only a single shape-from-texture method, and contained no measure of the correctness of the individual orientation constraints. Still, they showed the applicability of surface patch level fusion.

This surface patch level fusion approach, with one extension, will be used in the system that is being proposed. Now we can defined the more general patch which we call a *Texel Patch*:

> **Definition 3-1:** A *Texel Patch* is defined as a grouping of one or more pixels that are defined to be both components of the same surface and comprised of the same textural properties.

In other words, a texel patch is a surface patch that contains a single textural property. If the shape-from method generates orientation constraints measured for individual texels then the texel patch is equivalent to the texel. When the shape-from methods generate orientation constraints based on a measured change in a feature of the texture (e.g. the size of texels as in shape-from-uniform-texel-size) that can only be measured between groups of texels (such as texels in a window) then each group of texels will be considered a texel patch.

The fusion of multiple shape-from-texture cues will be performed at the texel patch level, based on the surface patch fusion model describe above. Texel

patch fusion has one major limitation. Since the constraints are fused per texel patch, a problem arises when shape-from cues are utilized which generate orientation constraints based on non-equivalent texel patch definitions (.e.g. texels as in shape-from-uniform-texel-size and edges as in shape-from-texel-isotropy).

This limitation can be removed by a multi-level fusion approach in which more than one texel patch definition is used. For each texel patch definition all of the shape-from-texture cues are applied to the texel patches and fused together. A surface reconstruction phase can be performed to build one or more surfaces out the different texel patches defined by the different texel patch definitions. The surface reconstruction and surface segmentation problems are discussed in Chapter 6.

For these and other reasons, an intermediate level has been chosen as the level at which orientation constraints are fused - the surface patch level. It has the advantage of having fewer elements than the pixel level (hundreds as compared to hundreds of thousands) without requiring additional a priori information about the surfaces.

3.2 The Augmented Texel

In this section a unified framework for fusing orientation constraints at the surface patch level is proposed. This approach is based on a new data structure, the *augmented texel*, in which orientation constraints generated by diverse textural cues are combined. Each augmented texel consists of a texel patch, orientation constraints for the texel patch, and a "correctness" weighting per constraint.

3.2.1 Texel Patch Definition

As described in the last chapter (see page 41), previous shape-from-texture cues can be classified in terms of their underlying assumptions and their measurement-unit (see definition 2.7). Since the measurement-unit is basic textural unit used in recovering orientation constraints, it should also be the level of abstraction at which orientation constraints should be fused.

The texel patch concept (see definition 3.1.3) is based on the observation that the generation of orientation constraints under perspective distortion is performed using one or more measurement-units (blobs, groups of edges, or heuristically set windows in the image, see figure 2-7). Each orientation constraint can be considered as defining the orientation of the individual texel, or groups of texels, that were used in calculating the constraint. If individual texels are used (as in blob-based cues or edge-based cues) then for each texel the grouping of neighboring pixels that constitutes the texel is defined as a texel patch. On the other hand if groups of texels (as in image window based cues) are used in calculating the orientation constraint then for each grouping of texels the image patch containing those texels is considered as a single texel patch. This definition of texel patches is intentionally loose. It allows for a range of texel patch definitions, based upon the way in which the texels are found and upon the specifics of the shape-from-texture methods used. It simplifies the way in which one class of shape-from-texture cues can be replaced by another class.

This definition allows a simple extension to the existing shape-from-texture methods beyond their current limitation of planar surfaces or simple non-planar surfaces based on a simple texture cue. Furthermore, it will allow a system based on this definition to fuse different types of shape-from methods. For example, it will allow shape-from-texture methods based on measured changes between blobs to be fused with each other (e.g. shape-from-uniform-texel-size with shape-from-uniform-texel-spacing) as well as shape-from-texture methods based on measured changes between edges to be fused with each other (e.g.

shape-from-texel-density and shape-from-texel-isotropy).

Since the measurement-unit is separated from tne orientation generation algorithm, the texel patch finding step can be considered as separate from the individual shape-from-texture methods. A single texel patch recovery algorithm can generate, as will be seen, texel patches for which many shape-from methods are applicable. In fact the same shape-from method is applicable to texel patches defined by a wide range of texel patch recovery algorithms. For example, if texels patches are recovered by a simple threshold-based blob-finding algorithm, then shape-from-uniform-texel-size can calculate changes in size between different blobs; if an edge-based definition is used, then the same shape-from-uniform-size cue can be used (it is then called a shape-from-uniform-density cue).

As stated in section 2.7 there are three basic image measurement-units: windows, edges, and blobs. The texel patch finding step can be defined to first find and then measure texel patches based on any of these definitions. In the next three subsections we will describe how the texel patch finding step can be defined based on each of these three measurement-units.

3.2.2 Window-Based Texel Patch Definitions

A window-based texel patch finder can be defined in any of a number of ways. The basic steps consist of:

- Carving the image into areas. This can be simply done by heuristically choosing a window size and separating the image into uniformly sized windows[11].
- Measuring the texel properties per window, e.g. counting the pixels above a threshold (here again the equivalence between shape-from-uniform-size and shape-from-density is shown).
- Suppling the texel properties to the shape-from-texture cues.

The specifics of each of these stages will not be described in this work.

---

[11]This is what most of the existing shape-from-texture cues that utilize windows do.

Rather they will be left for later research.

3.2.3 Edge-based Texel Patch Definitions

In an edge-based method the image is processed by applying any of the existing edge finders (e.g. [Lee, Pavlidis, and Huang 88a; Lee, Pavlidis, and Huang 88b]). Once the edges have been located they can either individually, or as small groups, be defined as texel patches. Since the edges have both size and position, shape-from-texel-size and shape-from-texel-spacing can be applied as well as the other shape-from-texture cues described in figure 2-8.

An edge-based definition has two problems. First, edges are more statistical in nature and therefore a large group of constraints will be required. Fortunately, an edge finding algorithm will generate many edges, and the application of any of the shape-from-texture cues that utilize more than one texel patch will create a large number of constraints.

Second, the number of texel patches that will be generated is large and thus require a substantial amount of processing power. One solution to the problem is to use a parallel processor where each processing element (or PE) finds the orientation of a single texel patch.

An interesting question that will not be further considered here is whether to define texel patches as a single edge or as a group of edges. If a group of edges is used and the groupings are equivalent in the area of the image they cover, then window-based shape-from-texture cues can be fused together with the edge-based cues.

3.2.4 Blob-based Texel Patch Definitions

In the system that will be described in the remainder of this chapter and the next chapter, the texel patch generator is defined as a simple bin thresholding-based, connected blob algorithm. As stated earlier, any of a wide range of texel patch generators could be used. This one has been selected because of its

simplicity; it shows the power of the fusion methodology rather than of the texel patch generator. A more complex texel patch generator would be used as part of a complete vision system.

In the method, the image is histogrammed, and the histogram peaks (a high concentration of pixels at that value) and valleys (a low concentration of pixels at that value) are recovered. A potential threshold range is defined to be any range of values containing a peak (a peak is heuristically defined as a single value with at least 1000 pixels) and delineated on each side by a valley. The change from a peak range to a valley range is heuristically set at value with 100 or less. These values were set based on an image size of approximately 512 x 512 pixels and were experimentally found to be effective.

The largest threshold range is ignored since it is assumed to contain the background; and the remaining threshold ranges are individually applied to the image.

Any four-connected blob with a minimum number of pixels is considered a texel patch. Currently the minimum size has been set at a soft threshold of 50 pixels; this limits the effects of noise by removing very small blobs. These very small blobs could be included by decreasing the threshold. But the effect would be a substantial increase in the computational costs in exchange for only a marginal increase in orientations for valid surface patches. (Further, tiny valid blobs will have high orientation and measurement errors due to their size.

This texel patch generator is simple and, as will be seen in Chapter 5, creates good results. This thesis does not purport to define the best blob finder or the best texel patch generator. More complex and "intelligent" texel patch recovery algorithms can be used and would increase the overall robustness and flexibility of the system.

3.2.5 Orientation Constraint Definitions

Shape-from-texture cues can generate either full orientations (e.g. shape-from-eccentricity) or orientation constraints (e.g. shape-from-uniform-texel-size, shape-from-uniform-texel-spacing, etc.), depending on the specifics of the shape-from-texture method. Potential orientations are stored within the *augmented texel* as (p,q) values, with a related correctness measure (defined in section 3.2.6). Potential orientation constraints are stored in the texel patch's *augmented texel* as vanishing points, which are mathematically equivalent to a class of other orientation notations (e.g. pan and tilt constraints) [Shafer, Kanade, and Kender 83; Kender 80]. Moreover, they are simple to generate and compact to store.

The generation of full orientations as well as orientation constraints might initially seem to cause difficulty in the fusion process. However, potential orientations are simply highly constrained orientation constraints. They are usable given a careful choice of the method used to integrate the different constraints. For the remainder of this chapter, unless specifically noted, full orientations will be considered as yet another class of orientation constraints.

In the next sub-section (section 3.3), existing shape-from-texture methods (shape-from-uniform-texel-spacing and shape-from-uniform-texel-size) as well as new shape-from-texture methods (shape-from-parallel-lines, shape-from-eccentricity, and shape-from-eccentricity-change) will be discussed, and the specifics of how they determine orientation constraints will be described.

3.2.6 Correctness Weighting Definition

Shape-from-texture cues generate orientation constraints based on a number of underlying assumptions about the image, the texture, the specific texels, etc. The fact that these assumptions are necessary has, in previous methods, been ignored; it is valid as long as the image and the textured surface it contains are limited to a small class in which the assumptions are valid.

In order to integrate a range of shape-from-texture cues, and to apply them

to a range of textured surfaces, the underlying assumptions of each shape-from-texture cue must be made explicit. In this work the underlying assumptions of the shape-from-texture cues are used in defining both the intra-cue and inter-cue weighting factors of the orientation constraints.

Shape-from-texture cues utilize two major types of assumptions. These assumptions can be classified as:

1. Constraint specific assumptions that are valid to differing degrees for the different constraints. (E.g., if the surface is curved then neighboring texels are more likely to be part of a planar sub-surface patch than texels separated by a greater distance).
2. General assumptions that relate equally to all of the cue's constraints and differentiate the constraints of one cue from another. (E.g. the assumption that texture is comprised of uniformly sized texels. The validity of this assumption affects all of the constraints generated by shape-from-texel-size equally.).

A correctness weighting factor W can be defined based on the two classes of assumptions, listed above. The assumptions of the first class are utilized in calculating an intra-cue correctness factor while the assumptions of the second class are used in defining an inter-cue normalization factor. The product of the two factors is an approximation to the likelihood that the constraint is correct. Individually each of the factors weights a different aspect of the orientation constraint as follows.

The intra-cue correctness factor I is defined for each shape-from-texture method, based upon underlying assumptions. Since no probabilistic formulation of whether an assumption holds for a specific image has yet been derived, we will try to approximate this result. We define I as:

$$I = \prod_{k=1}^{n} I_k \tag{3-1}$$

where n is the number of constraint-based assumptions that the shape-from-texture cue is based upon and I_k is the weighting factor from 0 to 1 that the

constraint fulfills the k^{th} assumptions. It is defined below for each of the shape-from-texture cues described. The effects of the weighting on the choice of the correct surface orientation is shown in Chapter 5.

The inter-cue normalization factor *W* serves two purposes. First, it distinguishes between incorrect constraints generated by non-valid cues and correct constraints generated by valid cues. Second, it also assures that no single shape-from cue gains ascendancy over other shape-from cues simply because the cue "naturally" generates more constraints. This normalization factor is generated separately for each shape-from cue, and then applied uniformly to all of the cue's constraints. The specifics of how the inter-cue normalization factor is generated is discussed in Chapter 4.

The use of correctness measures substantially extends the applicability of this approach beyond that of previous surface patch fusion systems, e.g. Ikeuchi's shape-from-apriori-texel-knowledge (section 2-2) and Aloimonos' shape-from-uniform-texel-size (section 2.3.3). It increases the system resilience to noise while allowing the system to deal with multiple textures and multiple surfaces (See examples in chapter 6). However, the current formulation is heuristic in nature. Future research may be able to ground the weightings as true probability distributions.

3.3 Shape-from Cues Revisited

In the remainder of this chapter, five different shape-from-texture methods are discussed: shape-from-uniform-texel-spacing, shape-from-uniform-texel-size, shape-from-texel-parallel-lines, and shape-from-eccentricity, shape-from-relative-eccentricity. The specific equations used are defined and the correctness weightings derived. The overall flow of the system as it has now been defined can be seen in figure 3-4.

To demonstrate the specific shape-from cues, an example image, shown in figure 3-5, has been created. This camera-acquired image consists of 12

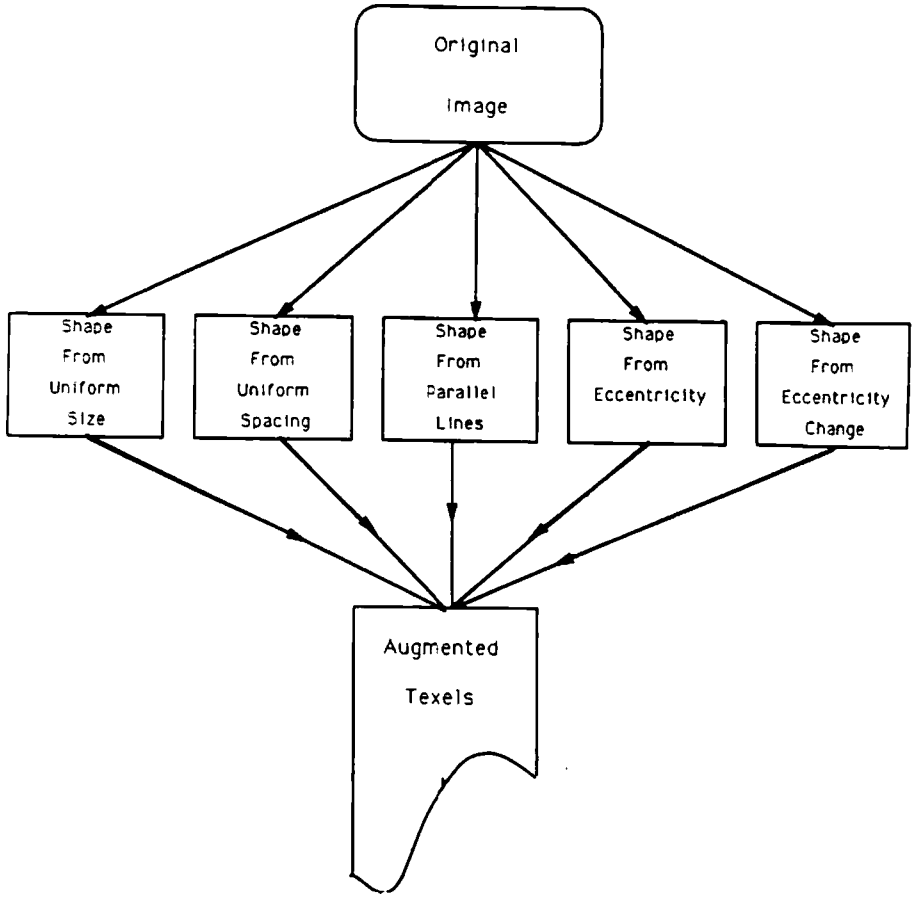


Figure 3-4: Constraint generation phase

uniformly spaced and uniformly sized circles. Because the image is both uniformly spaced and sized all of the five example shape-from cues are applicable to it.

To simply the example, three arbitrary texels, numbers 2,7, and 9 (see figure 3-6 for the texel numbering scheme), will be used to show the orientation constraints that each shape-from-texture cue generates. The texel numbering, shown in figure 3-6, was computer-generated. The texels are numbered in the order they are located in searching the image up from the lower left hand corner. Due to digitization errors, the texels are not uniformly positioned and this slight perturbation causes the system to number the texels in a "strange" manner.

It can not be stressed sufficiently that this image, as in all of the example imagery that will be used, is a real camera acquired image. Digitization errors as

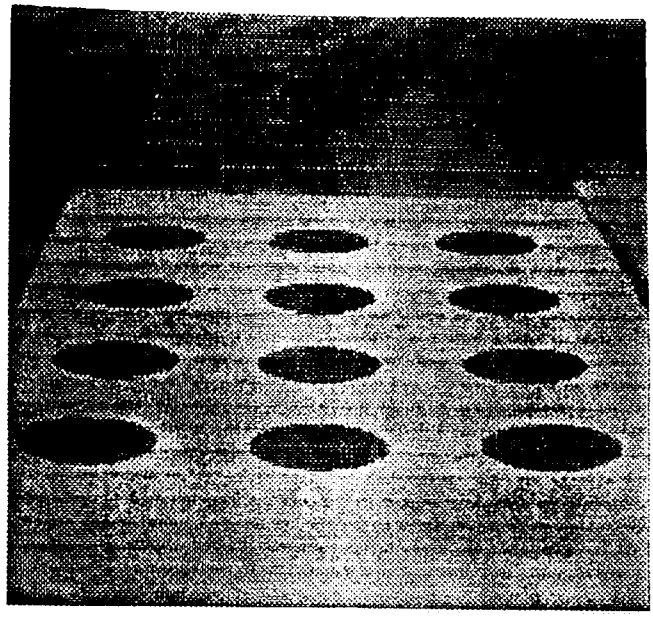


Figure 3-5: A texture of equal spaced and sized circles

can be seen in the slight perturbation of the placement of the texels are important and should not be ignored in testing the applicability of the shape-from cues. Therefore, throughout the use of this example various real world noises occur which the system must be able to deal with.

Orientation constraints will be shown throughout this work as vanishing point locations. A two dimensional coordinate system has been overlayed over the image with the origin located at the lower left hand corner of the image. Vanishing points are measured in terms of pixels from the coordinate system origin. The use of the pixel as the unit of measure rather than focal length distances is purely implementational and is used here to simplify the visualization of the orientations and distances.

3.3.1 Reference Points

Many of the shape-from-texture cues (e.g. shape-from-uniform-texel-spacing etc.) generate orientation constraints based on a measured change between texel patches. In computing this change a reference point on each texel patch must be used to measure the distance between texel patches. This point should fulfill three criteria:

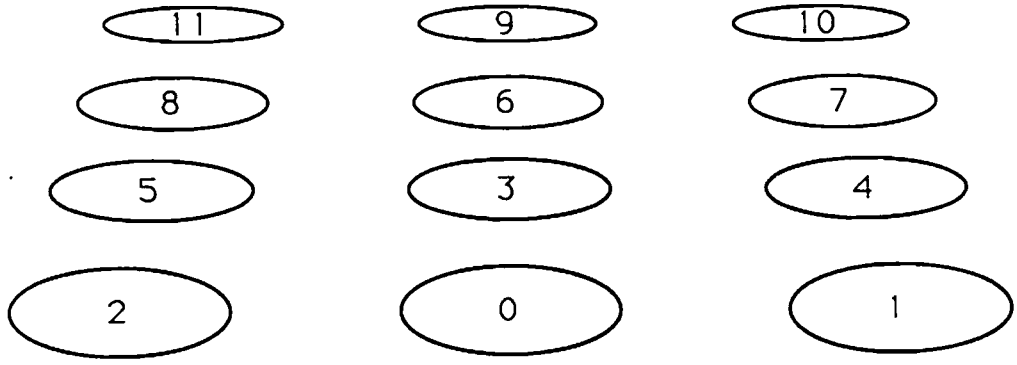


Figure 3-6: The numbering of texels from figure 3-5

Texel #	Row	Column	Size
0	109.502556	207.873825	2346
1	110.906601	362.931030	2334
2	112.206459	54.138805	2291
3	165.906235	208.555832	1621
4	166.914001	345.139435	1628
5	167.846725	72.239182	1618
6	211.410690	208.561768	1198
7	212.180969	330.709595	1188
8	212.786255	86.841576	1193
9	249.157837	208.556763	925
10	249.379242	319.078674	915
11	250.481247	98.292603	933

Figure 3-7: The center of mass and size of the texels of figure 3-5

1. Noise resistance - Real images contain different forms of spurious noise which can effectively add or subtract pixels from the texel patches found. Thus the reference points used should be chosen such that noise does not appreciably change the position of the reference points. This is important since even a single pixel shift in the position of the reference points can drastically change the orientation calculated.

2. Foreshortening and scaling consistency - The size and shape of texels change due to foreshortening and scaling effects. The reference points will not be totally unaffected. Rather the position of the reference points located on different texels should scale and foreshorten in exactly the same manner, if the texels are on the same surface patch.
3. Simplicity in calculation - For obvious reasons simplicity should always be a guide in choosing among equivalent options.

A number of different points were considered as potential reference points including edge points, corners, and the center of mass. The center of mass has been chosen since it seems to best fulfill the three criteria listed above as follows:

1. Since the point is an average, based on the whole texel, small quantities of noise create substantially smaller shifts in the placement of the center of mass than it would for many other reference points, e.g. edge- and corner-based points can change drastically if a single noise point falls on the edge or corner.
2. The center of mass scales approximately uniformly for all texels on the same surface patch. Texels foreshorten differently based on their placement on the surface. When the texels are small in relationship to the focal length, as would seem to occur in most imaging systems, foreshortening compression effects appear to be no greater than other errors (e.g. the effects of digitization.).
3. The center of mass can be calculated during the texel patch finding stage, described in section 3.2.1 with no measurable change in the speed of the algorithm.

The use of the center of mass does not preclude the use of additional reference points. Other noise resistant points could be used to gain additional robustness and might be necessitated by other definitions of "texel patch".

3.3.2 Shape-From-Uniform-Texel-Spacing

As discussed in section 2.2, shape-from-uniform-texel-spacing derives orientation constraints based on the assumption that the texels are uniformly spaced. The method takes three texels and derives an orientation constraint based on the change in spacing between the three texels (see figure 3-8) . The only major differences between the shape-from-uniform-texel-spacing algorithm

implemented in this multiple shape-from-texture system and previous implementations (e.g. [Moerdler and Kender 85]) are the reference points used to measure spacing changes (the center of mass), and the how texels are defined.

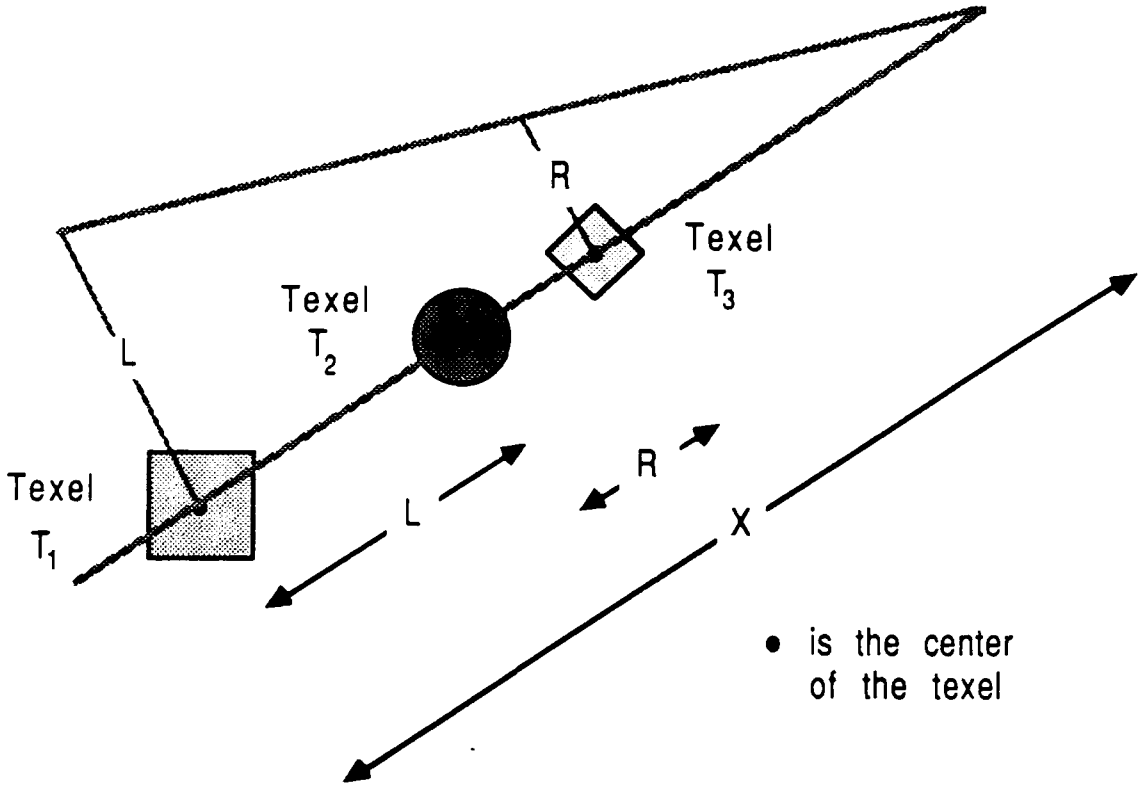


Figure 3-8: A geometrical representation of shape-from-uniform-texel-spacing.

Since no a priori knowledge about the surface, texture, or number of textures can be made the shape-from-uniform-texel-spacing module considers every possible grouping of three texel patches. Only those groupings whose reference points are approximately co-linear (defined below) are utilized in generating orientation constraints. An approximation to co-linearity is used to increase the system's resistance to errors in reference point position.

3.3.2.1 The Calculation

Given the location of the reference points of any three co-linear texels T_1, T_2, and T_3 (see figure 3-8) an orientation constrain can be generated similar to that described in section 2.2. In the calculation, two simplifications that were previously utilized in the literature are made explicit here. The distance between texels T_1 and T_2 is not assumed to be greater than the distance between texels T_2 and T_3, nor is it assumed that there exist no other texels located among the three texels.

Equation 2-1 can now be reformulated, in two steps, to derive orientation constraints for any three co-linear texels. In the simple case the three texels are chosen in a row on an imaginary image line that connects the three texels. The distance between the center of mass of texel T_1 and T_2 is defined as **L** and the distance between the center of mass of texel T_2 and texel T_3 is defined as **R**. A vanishing point **V**, which defines a constraint on the orientation of the three texels, is found at a distance **X** from T_1 measured on the line connecting the three texels:

$$X = \frac{(1+R) \times (L+R)}{L-R} \quad if\, L > R$$

$$X = \frac{L \times (L+R)}{L-R} \qquad if\, L < R$$

For the more general case, other texels may be located between the three texels. The number of such texels must be taken into consideration in generating the orientation constraints. Therefore, if the number of texels located between texels T_1 and T_2 is N_L and the number of texels between texels T_2 and T_3 is N_R then the distances **R** and **L** can be redefined as $\mathbf{R_N} = \frac{R}{N_R+1}$ and $\mathbf{L_N} = \frac{L}{N_L+1}$ and the relative distance X from texel T_1 is defined as:

$$X = \frac{(1 + R_N) \times (L + R)}{L_N - R_N} \quad if\ L_N > R_N$$

$$X = \frac{L_N \times (L + R)}{L_N - R_N} \quad if\ L_N < R_N \qquad (3\text{–}2)$$

From the relative distance X a specific vanishing point can be determined. The (x,y) coordinates of the vanishing point **V** in terms of the relative distance X given in equation 3-2 are:

$$V_x = T_{1_x} + \frac{X}{L + R} \times (T_{2_x} - T_{1_x}) \qquad (3\text{–}3)$$

$$V_y = T_{1_y} + \frac{X}{L + R} \times (T_{2_y} - T_{1_y})$$

To better understand the cue let us consider texel number 7 of the example image of figure 3-5 (texel number is shown in figure 3-6). Texel number 7 is co-linear to 5 groups of texels (1,4,7) (1,7,10) (2,3,7) (4,7,10) (6,7,8) (see figure 3-1) thus creating 5 orientation constraints for texel number 7. All five of these orientation constraints will indicate the same orientation.

The last constraint was generated by texels (6,7,8) which are approximately uniformly spaced on the image. Thus a vanishing point was generated along the approximately horizontal line connecting the three texels at a column that approaches infinity. The vanishing point is not exactly at the same row as the three texels since the centers of mass of the texels are not all on the same row.

The constraints do not define exactly the same orientation because of the effects of noise and digitization errors. This fact is one of the reasons that an integrated approach is important.

Texel # 2			Texel # 7			Texel # 9		
V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used
609.56	1430.03	3 7	639.73	194.68	1 4	642.42	210.48	0 3
634.94	224.10	5 8	630.40	198.41	1 10	648.28	210.51	0 6
644.89	224.24	5 11	609.56	1430.03	2 3	654.26	208.56	3 6
654.431	227.28	8 11	625.86	200.10	4 10	736.50	-97275.57	10 11
			387.26	-70209.52	6 8			

Table 3-1: The vanishing points created by shape-from-uniform-texel-spacing for texels # 2, 7 and 9 of figure 3-5

3.3.2.2 The Weighting

Once the vanishing point is generated and stored as an orientation constraint for each of the three texels, an intra-cue correctness weighting must be generated. This weighting is defined utilizing the underlying assumptions of shape-from-uniform-texel-spacing: uniform spacing, planarity, and co-linearity.

The major assumption of shape-from-uniform-texel-spacing is that the texels are uniformly spaced in the surface. This assumption is utilized in generating the inter-cue weighting factor which will be discussed in section 4.3, but is not used for inter-cue normalization.

The next two assumptions, planarity and co-linearity, are based on characteristics of the texels and of the surface patch surrounding the texels that are used in generating each specific orientation constraint. Each of these assumptions will define a component of the intra-cue weight.

The first factor reflects the likelihood of the three texels fulfilling the planarity assumption. Since no a priori knowledge exists about the planarity or curvature of the surface, it is impossible to define exactly what the probability is that two or more specific texels lie on a planar surface patch. Instead, a planarity

"confidence" can be defined by a simple, linearly decreasing function based on the Euclidean distance between the texels.

When distance approaches the maximum distance in the image the confidence should approach zero. If the Euclidean distance between the centers of mass of two texel patches i and j is defined as $D_{i,j}$ and the maximum Euclidean distance in the image[12] is defined as E then one general formulation of a planarity "confidence" function P can be:

$$P = 1.0 - \frac{D_{i,j}}{E} \tag{3–4}$$

For shape-from-uniform-texel-spacing three texel patches must all lie on a planar surface patch which must be at least as large as the distance between the farthest separated texels (texels T_1 and T_3). Therefore the weighting factor due to the planarity assumption P for shape-from-uniform-texel-spacing is simply:

$$P = 1.0 - \frac{D_{1,3}}{E}$$

The second assumption is that the texels are co-linear on the surface. This assumption can be incorrect if the imaging geometry combines with the textural properties and the effects of noise to create false co-linear texels in the image. These false placements are the cause of certain optical illusions for the human vision system.

The first factor can create incorrect constraints. As long as nature does not mimic perspective distortion, these algorithms will not create a sufficient number of compatible constraints to affect the orientation found. The second factor can cause groupings of three texels to be ignored and therefore no orientation constraints to be generated. This can occur in a perfect image in which

---

[12]In a 512 x 512 image $E = \sqrt{512^2 + 512^2} = 724.0773$.

digitization and noise effects are minuscule and the surface is highly curved and the groupings cross the curvature (see figure 3-9). In this instance the surface planarity assumption has failed sufficiently that even if constraints were created they would be wrong.

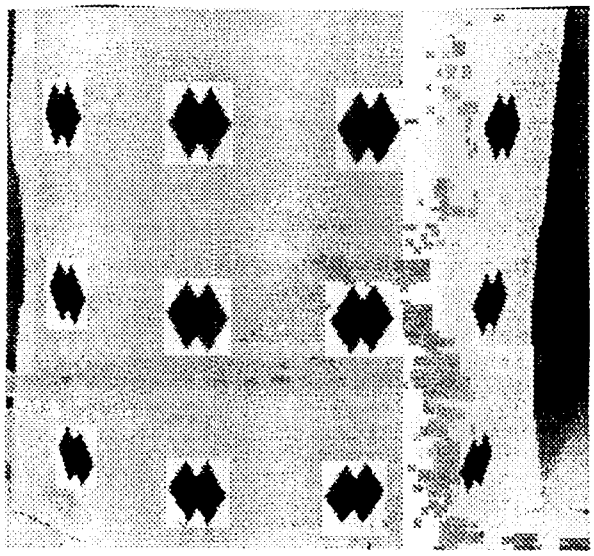


Figure 3-9: A highly curved textured surface

Digitization errors and the effects of noise perturb texel placement information, as can be be seen in the example image of figure 3-5. Shape-from-uniform-texel-spacing will generate either five constraints for some of the texels and only four constraints for other texels.

Digitization errors and the effects of noise cause the texel patch generator to either add additional or remove pixels from the texels. These slight perturbations affect the center of mass calculations, and thus cause co-linear texels to be calculated as not co-linear. This can been seen in the example in that texels 0,1, and 2 are not computed as co-linear and thus no orientation constraint is generated by shape-from-spacing for these three texels (see the first column of figure 3-1.

The co-linearity component of the intra-cue weighting is generated by

calculating the degree to which the three texels T_1, T_2, and T_3 are co-linear. This consists of determining the line that connects texels T_1 and T_2 and calculating the closest point on the line to texel T_3. As the distance from texel T_3 to the line increases confidence decreases. When the distance reaches 4 pixels (a heuristically set maximum distance) it reaches zero.

3.3.3 Shape-From-Size

The second shape-from-texel method that is used in the multiple shape-from-texture system is shape-from-uniform-texel-size. As described in section 2.3 shape-from-uniform-texel-size assumes that the texels are of uniform size prior to the effects of perspective distortion. Given this assumption from the size of two texels and the spacing between the texels an orientation constraint can be determined.

The use of the center of mass as the reference point in shape-from-uniform-texel-size is not new to this work. Both Ohta [Ohta et. al. 81] and Aloimonos [Aloimonos and Swain 85] used it although without justification. However, they recognized the need for noise resistance, foreshortening and scaling consistency, and ease of calculation.

This work attempts to assume as little possible about the surfaces and textures. Since no a priori criteria seems to be available for the selection of groupings of two texels all groupings of two texels will be used. Groupings of texels that contain texels from different surfaces, different textures, or noise will create incorrect orientation constraints; these should be ignored in the relaxation step. This approach differs from that of previous work which assumes only a single texture and a single surface.

3.3.3.1 The Calculations

The orientation constraint calculation method can now be specified for shape-from-uniform-texel-size. For each texel T_i the system computes the area of the texel S_i found by thresholding and connectivity analysis.

The method generates all combinations of two texels, T_1 and T_2, and calculates for each a vanishing point. This vanishing point *V* is located at a distance *D* from the reference point of texel T_1, along the line that connects the reference point of texel T_1 and T_2:

$$D = \frac{(1 + S_2^{\frac{1}{3}}) \times E}{S_1^{\frac{1}{3}} - S_2^{\frac{1}{3}}} \quad if\ S_1^{\frac{1}{3}} > S_2^{\frac{1}{3}}$$

$$D = \frac{S_1^{\frac{1}{3}} \times E}{S_1^{\frac{1}{3}} - S_2^{\frac{1}{3}}} \quad if\ S_1^{\frac{1}{3}} < S_2^{\frac{1}{3}} \qquad (3\text{–}5)$$

where E is the Euclidean distance from the reference point of texel T_1 to the reference point of texel texel T_2 (see figure 3-10).

The orientation constraint, defined as the coordinates of the vanishing point, can be calculated from the equation:

$$V_x = T_{1_x} + \frac{D}{E} \times (T_{2_x} - T_{1_x})$$

$$V_y = T_{1_y} + \frac{D}{E} \times (T_{2_y} - T_{1_y}) \qquad (3\text{–}6)$$

Both the vanishing point distance *D* and the vanishing point location V_x, V_y

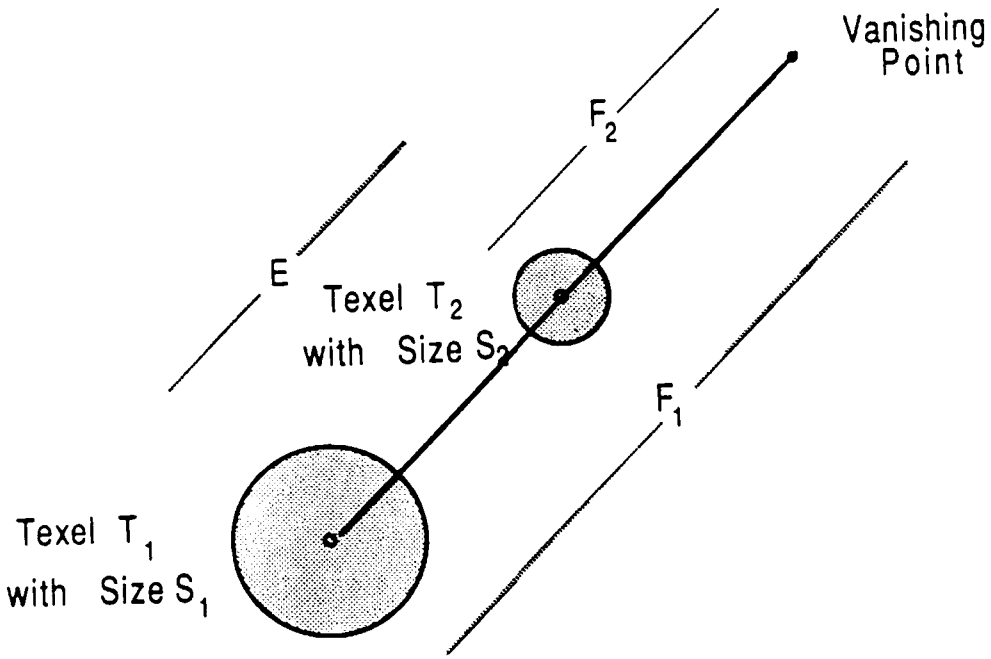


Figure 3-10: How texel size can be used to derive orientation calculations are very similar to those of shape-from-uniform-texel-spacing.

Now let us consider how the cue acts with real camera acquired imagery. Given the test example image of figure 3-5, shape-from-uniform-texel-size will generate 11 orientation constraints for each of the 12 texels (orientation constraints, defined as vanishing points, are shown in figure 3-2). Nine of the eleven constraints cluster around the same column for each of the three texels. As stated in the previous subsection, the orientation constraints created by vertically aligned texels do not fall on the same row because of the effects of noise and digitization errors.

The remaining two constraints are located at "infinitely" distant points horizontal to the vanishing line. These points are "infinitely" far away because the surface has little or no q orientation.

Texel # 2			Texel # 7			Texel # 9		
V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used
452.79	-19310.09	0	615.48	813.18	0	633.11	210.44	0
321.27	-49610.11	1	613.34	203.08	1	631.70	-218.60	1
605.25	1471.92	3	620.71	1460.89	2	637.13	646.01	2
620.48	2757.76	4	636.15	1449.88	3	654.02	208.56	3
620.50	219.49	5	620.94	200.41	4	645.78	-450.11	4
622.63	848.68	6	620.93	2713.77	5	646.02	873.89	5
620.71	1460.89	7	487.47	43986.45	6	668.43	208.50	6
626.74	221.44	8	-219.88	174407.08	8	674.24	-1195.69	7
637.13	646.01	9	674.24	-1195.69	9	660.08	1583.70	8
632.64	1059.34	10	658.45	191.17	10	310.38	30767.34	10
646.55	224.77	11	707.13	-2672.78	11	-211.21	38565.52	11

Table 3-2: The vanishing points generated by shape-from-uniform-texel-size for texels # 2, 7 and 9 of figure 3-5

3.3.3.2 The Weighting

For each orientation constraint calculated, shape-from-uniform-texel-size must also generate a correctness weighting. This weighting is similar to that of shape-from-uniform-texel-spacing. Shape-from-uniform-texel-size uses two major assumptions about the texture and the surface: surface patch planarity, and texel size uniformity prior to the effects of perspective distortion.

The same planarity "confidence" function as developed for shape-from-uniform-texel-spacing can be used here except that only two texels need to be part of the same planar patch. As will be shown in chapter 5 the experimental evidence backs the use of this simple function.

The most important underlying assumption is that the texels are uniformly sized. This assumption, as is the assumption that the texels are uniformly spaced for shape-from-uniform-texel-spacing, is built into the inter-cue normalization factor (see section 4.3). To summarize, if the texels are not of uniform size on the original surface, then shape-from-uniform-texel-size will generate conflicting constraints which will act as noise in the relaxation step. Other, valid constraints

will define an orientation of greater prominence.

3.3.4 Shape-from-relative-eccentricity

This third shape-from-texture cue is new, derived during this research, by the method of section 2.7. Shape-from-uniform-size can be modified by replacing the texel size measure by the eccentricity measure of the texel.

3.3.4.1 The Calculations

Eccentricity is the proportion of the minor to major axis of symmetry (see figure 3-11). An approximation to the eccentricity measure can be simply calculated by any of of number of methods (e.g. [Ballard and Brown 82; Brown and Shvaytser 88]) which are based on the moments of the blob. Specifically, given a blob **B** whose size is *n* pixels and whose center is located at $(\mathbf{x_0},\mathbf{y_0})$ the ij^{th} moments $\mathbf{M_{ij}}$ are:

$$M_{ij} = \sum_{x \text{ in } B} (xy_0 - x)^i (y_0 - y)^j$$

The approximate eccentricity *e* is then determined from the x and y its components.

$$e_x = M_{20} + M_{02}$$

$$e_y = \sqrt{(M_{20} - M_{02})^2 + (2\,M_{11})^2}$$

$$e = \frac{e_x - e_y}{e_x + e_y} \qquad (3\text{–}7)$$

As an oriented object moves along a surface away from the viewer the eccentricity measure decreases. Given two circles C_1 and C_2, their eccentricity measures can be related mathematically to the surface orientation in terms of p and q. In the simplified case where p=0, the eccentricities are:

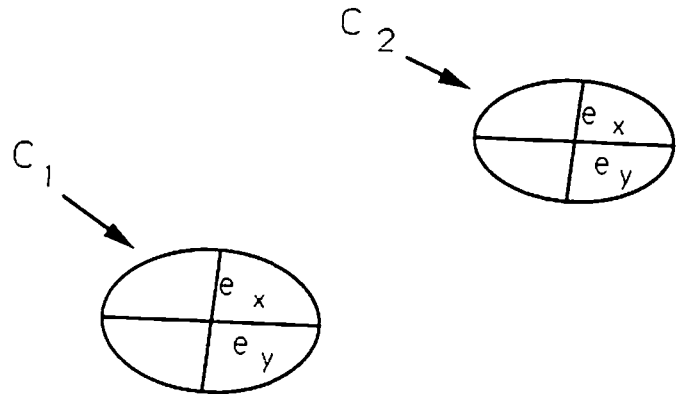


Figure 3-11: Measuring the eccentricity of a blob

$$E_1 = \frac{1-q\,y_1}{\sqrt{1+q^2}}$$

$$E_2 = \frac{1-q\,y_2}{\sqrt{1+q^2}}$$

where y_1 and y_2 are the distances from the center of circles C_1 and C_2 respectively to the vanishing line.

The orientation value q can be recovered from:

$$q = \frac{E_1 - E_2}{E_1 y_2 - E_2 y_1}$$

This result is similar to that of shape-from-uniform-texel-size. The general case for $p \neq 0$ can be calculated as in shape-from-uniform-texel-size. (For the complete solution see [Kender and Moerdler rt]) We can substitute the eccentricity measure E_1 and E_2 of two texels T_1 and T_2 into the shape-from-

uniform-size equation of 3-5 as follows:

$$D = \frac{(1 + E_2^{\frac{1}{3}}) \times E}{E_1^{\frac{1}{3}} - E_2^{\frac{1}{3}}} \quad if\ E_1^{\frac{1}{3}} > E_2^{\frac{1}{3}}$$

$$D = \frac{E_1^{\frac{1}{3}} \times E}{E_1^{\frac{1}{3}} - E_2^{\frac{1}{3}}} \quad if\ E_1^{\frac{1}{3}} < E_2^{\frac{1}{3}} \qquad (3\text{–}8)$$

where E is the Euclidean distance distance between the centers of mass of the texels. The vanishing point location can then be found directly from equation 3-6.

$$V_x = T_{1_x} + \frac{D}{E} \times (T_{2_x} - T_{1_x})$$

3–6

$$V_y = T_{1_y} + \frac{D}{E} \times (T_{2_y} - T_{1_y})$$

3.3.4.2 The Weighting

The inter-cue weighting of the shape-from-relative-eccentricity cue is generated in the same way as the weighting of the shape-from-uniform-texel-size cue. The weighting is based on the distance between the texels and utilizes the same planarity function.

Texel # 2			Texel # 7			Texel # 9		
V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used	V P 'Row	V P Col	Texels Used
4923.05	-274058.41	0	687.84	899.78	0	674.95	210.39	0
-19.37	31628.35	1	710.30	172.43	1	689.06	-282.89	1
623.12	1524.43	3	676.81	1616.42	2	667.67	680.50	2
632.25	2821.41	4	741.29	1726.90	3	692.60	208.15	3
700.27	246.16	5	730.46	165.26	4	687.53	-520.11	4
654.96	899.42	6	654.65	2910.14	5	652.37	884.31	5
676.81	1616.42	7	98.54	7364.79	6	695.11	207.81	6
702.23	246.42	8	138.59	30622.40	8	650.12	117.41	7
667.67	680.50	9	650.12	117.41	9	609.33	1413.28	8
686.17	1162.68	10	706.54	175.48	10	224.62	867.76	10
718.85	248.20	11	831.93	-3431.99	11	195.81	4666.35	11

Table 3-3: The vanishing points generated by shape-from-relative-eccentricity for texels # 2, 7 and 9 of figure 3-5

3.3.5 Shape-from-Eccentricity

The fourth shape-from-texture cue, also new to this work, is similar to the texel isotropy assumption of section 2.4. In that section the texture is assumed to have a uniform isotropy. The change in isotropy is measured in terms of edge orientations and used to determine the surface orientation.

In this new method groups of edges are replaced by a single texel patch and the isotropy is measured in terms of the eccentricity measure. By assuming that the major and minor axis of symmetry of the texel patch are equal on the surface, we are effectively assuming that the orientation of the texel patch's edges are uniformly distributed. In the optimal case, in which the texel patch on the surface is a circle, these are equivalent. In the less optimal case (e.g. squares) the method continues to work but not as well.

The similarity between the new method and previous shape-from-isotropy cues can be seen in the use of only a single texel patch to recover orientation. Other shape-from-texture cues (e.g. shape-from-uniform-size) utilize measured changes between patches.

3.3.5.1 The Calculations

The method is based on the foreshortening compression effect on blobs whose two moments are equal. The most obvious case is that of circles. A circle under perspective distortion (which is approximated by assuming local orthography) appears as an ellipse. Since this method is new, created during this research, the derivation of the method will be discussed. For further information on the derivation see [Kender and Moerdler rt].

The validity of the method is derived as follows for the case of surfaces where p = 0. (The general case follows from simple rotations of the image coordinate system) Given the knowledge that:

1. The ellipse goes through the points (±w,0) and (0,±h).
2. The tangent at (0,±h) = 0.

and the notational conventions shown in figure 3-12, we can derive the orientation of the surface. The first step is to take the standard equation for an ellipse:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

We can substitute D = E = 0 since the center is the origin. Since (±w,0) is on the ellipse this gives us:

$$Aw^2 + F = 0, \text{fixing F in terms of A and w.}$$

Now, after implicit differentiation the equation becomes:

$$2Ax + Bxy' + By + 2Cyy' = 0$$

Using the fact that the tangent at (±d,±h) = 0, results in:

$$B = \frac{-2Ad}{h}$$

Next, we apply the knowledge that (±d,±h) is on the ellipse to determine that:

$$C = A\frac{D^2 + w^2}{h^2}$$

which can be substituted into the equation for the circles to derive that:

$$x^2 - \frac{2d}{h}xy + \frac{d^2 + w^2}{h^2}y^2 - w^2 = 0$$

This can be checked by calculating the tangent at (±w,0) since we know that:

the tangent at (±w,0) = h/d.

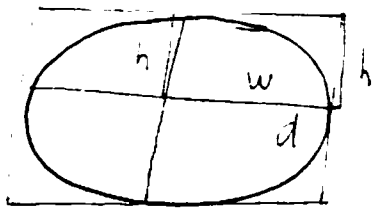


Figure 3-12: Measuring the axis of an ellipse.

Next we derive a method to calculate the orientation of a surface given an ellipse.

First calculate the angle of orientation σ of a ellipse in its plane:

$$cot2\sigma = \frac{A - C}{B} = \frac{d^2 + w^2 + h^2}{2dh}$$

For small σ,

$$\tan2\sigma = 2\tan\sigma,$$

or

$$tan\sigma = \frac{dh}{d^2 + w^2 + h^2}$$

Note that σ is measurable and that ε (the eccentricity) is also measurable, and approximately equal to h/w.

We next relate the perspective distortions of the circle to the ellipse's position (x,y) in the image, and the surface orientation (0,q). We adopt a kind of paraperspective, in which the ellipse is considered bounded by a perfect parallelogram.

By reasoning similar to that used for lengths under perspective, it can be shown that the changes in w obey a straightforward foreshortening, inversely proportional to their distance from the vanishing line. Thus, w=k(1-qy) where k is a scale factor for the ellipse. Since d measures the "ellipse skew", it is proportional to the amount of offset from the y axis, and inversely proportional to the distance from the vanishing line. Thus, it can be shown that:

$$d = \frac{kxh}{(y - 1/q)}$$

Lastly, h obeys the foreshortening typical of those lengths oriented perpendicular to the vanishing line, and is :

$$h = \frac{(1 - qy)^2}{\sqrt{1 + q^2}}$$

$$w = 1 - qy$$

$$d = \frac{xh}{y - 1/q}$$

Therefore finding all ellipse parameters in terms of the image-forming parameters (x,y) and q, we find:

$$tan2\sigma = \frac{2xb}{x^2 + R^2 - b^2} \qquad (3\text{–}9)$$

where:

$$b = \frac{1}{q} - y$$

$$R = \frac{\sqrt{1 + q^2}}{q}$$

If we let $\varepsilon = \frac{b}{R}$, in imperfect analogy to $\frac{h}{w}$ we can proceed:

Note that R is independent of x. Thus the eccentricity can be expressed independently of x as:

$$\varepsilon = (1 - qy)/\sqrt{1 + q^2}$$

From this, by the quadractic formula, we can relate q to y and ε.

$$q = y \pm \varepsilon \frac{\sqrt{y^2 - \varepsilon^2 + 1}}{y^2 - \varepsilon^2} \qquad (3\text{–}10)$$

Now we have two non-linear equations (3-9 and 3-10) that relate the image observables (x,y), ε, and σ to the parameter we wish to obtain, q. Unfortunately, there does not appear to be a simple closed form solution. Equation 3-9 expresses σ = f(x,y,q) and equation 3-10 expresses q = g(y,ε). However, they can be iteratively solved by a predictor-corrector method as follows:

STEP 1: Estimate q using ε and y in equation 3-10. This assumes that the effect of ellipse skew is zero and depends on the analogy that ε = h/w. In other words, σ is assumed to be zero.

STEP 2: Refine the estimate by calculating σ from equation 3-9. This estimates how much ellipse skew there ought to have been at (x,y) for the value of q to be exact.

STEP 3: Rotate coordinate so that now the ellipse does infact skew at the value calculated in step 2. Refine q using equation 3-10, in this new coordinate system.

Step 2 and 3 are repeated until successive estimates are within a tolerance. Three iterations on many images gave answers within five decimal digits of precision. Note that further accuracy is unnecessary (and deceptive) due to the noise in calculating (x,y) and ε anyway. Calculations for the general case of $p \neq 0$ are similar, except for the overall change of the coordinate system to map the general problem into the p=0 case.

Texel # 2			Texel # 7			Texel # 9		
p value	q value	Texels	p value	q value	Texels	p value	q value	Texels
-0.132360	-1.973796		0.117001	-3.073282		-0.044071	-3.870390	
0.319656	4.370194		-0.148062	3.814801		0.041297	3.631514	

Table 3-4: The vanishing points generated for texels # 0,7 and 9 for figure 3-5

3.3.5.2 The Weighting

Shape-from-eccentricity generates two orientations of each texel, the potential orientations of the visible and invisible surfaces of the surface patch. Since no a priori information exists as to the correct orientation, the cue can not choose between the two and therefore supplies both orientations to the fusion stage.

Since both orientations are equally likely of being correct there is no inter-cue weighting of the shape-from-eccentricity cue, both orientations are equally

weighted.

3.3.6 Shape-From-Parallel-Lines

The fifth shape-from-texture method is shape-from-parallel-lines. The major assumption of this cue is that the surface texture is comprised of parallel groupings, most often lines. Previous use of this assumption consisted of Kender's [Kender 80] theoretic application of parallel lines to derive shape and Magee and Aggarwal [Magee 84] work which generated vanishing points given parallel image line segments.

Limiting the texels to consisting of parallel lines is a strong assumption about the texels. The shape-from-parallel-lines method that is implemented here uses the weaker assumption that the texels are located on parallel virtual lines. This variation allows for a wider range of textured surfaces, and subsumes the use of parallel line texels themselves.

A virtual line is defined as the image line connecting the center of mass of two or more image texels. Other virtual line definitions are possible and would generate additional orientation constraints. The center of mass is used for generating the virtual lines for the same reasons that the center of mass was used to measure texel spacing distances in shape-from-uniform-texel-spacing (see page 60), namely it scales approximately uniformly for all texels on the same surface patch and it is resilient to single pixel noise since it is an average value.

Since the virtual lines that connect the texels are assumed to be parallel on the surface, the lack of parallelism in the image constrains the surface orientation. The point at which the virtual lines converge is a vanishing point.

This assumption is related to the uniform spacing assumption of shape-from-uniform-texel-spacing, and often if one is valid so is the other. Yet the two cues cover non-overlapping sets of image textures. Including both methods in

the multiple shape-from-texture system has a number of advantages:

1. Shape-from-parallel-lines requires a minimum of four texels, two on each of two virtual lines in order to determine a single constraint. Shape-from-uniform-texel-spacing requires only three texels on one line.

3.3.6.1 The Calculation

Orientation constraints are generated by first determining all virtual lines connecting pairs of texels, and then calculating for each set of two lines the point of intersection of the lines. This creates a large number of constraints, most of which are incorrect. In fact, if there are *n* texels there would be $O(n^2)$ virtual lines and $O(n^4)$ constraints generated.

In order to decrease the overall number of constraints generated, two simple heuristic limitations are imposed on the choice of virtual lines:

1. Two texels T_a and T_b are used to generate a virtual line if and only if there exists a third texel T_c whose center of mass is approximately co-linear with the center of mass of T_a and T_b. Approximate co-linearity is defined exactly the same way as it was previously defined for shape-from-uniform-texel-spacing (see section 3.3.2.2, page 68).
2. For the virtual lines to be parallel on the surface the point of intersection of the lines can not be located on the convex hull bounded by the centers of mass of the texels. Therefore, prior to storing any constraint the intersection point is checked to assure that it is not located between texels T_1 and T_2 or between texels T_3 and T_4 (see figure 3-12 for examples of legal and illegal virtual lines).

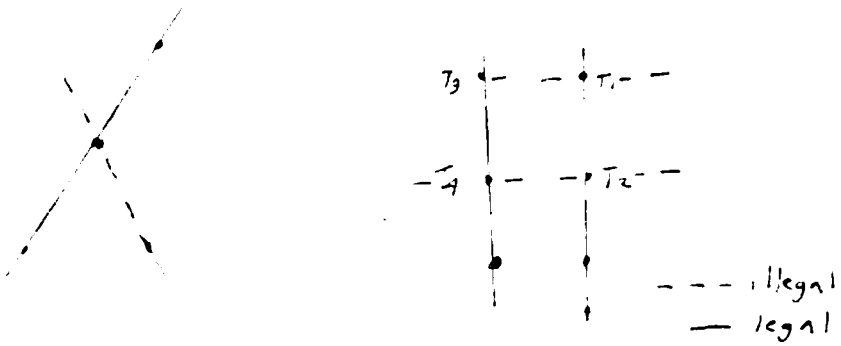


Figure 3-13: Examples of legal and illegal virtual line segments

The equations for generating orientation constraints given two virtual lines $L_{1,2}$ and $L_{3,4}$ are derived based on the standard geometric equations for the intersection of two lines.

3.3.6.2 The Weighting

Once the orientation constraints have been generated it is necessary to formulate the intra-cue correctness measure. Shape-from-parallel-lines is based upon two underlying assumptions: the texels are located on parallel lines, and that each set of texels is located on a planar surface patch. The first assumption of parallelism is handled in the inter-cue normalization factor.

The planarity assumption is similar to that of the previous shape-from-texture methods. For the shape-from-texture method to be applicable the texels must be located on planar surface patches. Texels T_1 and T_2 must be located on a line which is parallel on the surface to that of texels T_3 and T_4.

Initially one might assume that all four texels must be co-planar which would constrain the images for which this shape-from-texture method would be applicable. Fortunately, this four way co-planarity can be replaced with the less stringent assumption of texels T_1 and T_2 being co-planar and texels T_3 and T_4 being co-planar. It is important to stress that this is texel co-planarity not simply point planarity; the texels themselves are planar blobs.

In order to understand why this heuristically based simplification is acceptable let us consider the possible surface curvature conditions in which the method is and is not applicable:

1. The surface patch on which the four texels are located is planar. In this situation the method is obviously applicable and there exist no difference between assuming that the four texels are co-planar and that each set of two texels is co-planar.
2. The surface has one principal axis of curvature equal to zero and texels T_1 and T_2 are co-planar on the curved surface and texels T_3 and T_4 are also co-planar on the curved surface. The virtual line connecting T_1 and T_2 is parallel on the surface to the virtual line

connecting T_3 and T_4 and both are parallel to the direction of zero curvature. Under these conditions the lack of parallelism in the image can be used to generate an orientation constraint as described earlier. Assuming that the four texels are co-planar is unnecessary in this situation since the method is able to generate correct constraints even though the four texels are not co-planar.

3. The surface has one principal axis of curvature equal to zero but the virtual line connecting texels T_1 and T_2 and/or the virtual line connecting texels T_3 and T_4 are not parallel to the direction of zero curvature . If all sets of two texels were utilized to generate virtual lines then this would generate an incorrect constraint. For this reason each set of two texels that were chosen to generate virtual lines had to be approximately co-linear with a third texel. If the virtual line has curvature on the surface, then the chances that three texels in the image would be approximately co-linear is small and this choice of texels would not generate constraints. But again in this situation the more stringent assumption of four texel co-planarity is unnecessary.

4. The surface has two non zero curvatures. When the curvature is small in relationship to the texel spacing then this problem is handled as in the previous situations. When the curvature is large the chances that sets of three texels would be co-linear in the image is small and few if any constraints would be generated.

For the reasons described above the system need only compute a likelihood that texels T_1 and T_2 are co-planar and that texel T_3 and T_4 are co-planar. The inter-cue weighting I is the product of two components I_1 and I_2 where I_1 is the planarity "confidence" $P_{1,2}$ (see equation 3-4 page 67) that texels T_1 and T_2 are co-planar and I_2 is the planarity $P_{3,4}$ that texels T_3 and T_4 are co-planar. This planarity function is heuristically chosen to decrease as the square of the inter-texel distance.

3.4 Summary

Chapter 2 described a number of different shape-from-texture cues, each of which were limited in their applicability due to the underlying assumptions of the cue. In order to create a more general shape-from-texture system based on the integration of multiple cues, a level of abstraction at which to fuses the cues must be chosen, and existing shape-from-texture cues must be modified to generate

Texel # 2			Texel # 7			Texel # 9		
V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used
603.13	213.84	0 3 5	580.37	213.57	0 3 1	591.60	210.23	0 1 4
603.40	213.85	0 3 8	579.69	213.56	0 3 4	590.86	210.23	0 1 7
612.71	213.96	0 3 11	586.59	213.64	0 3 10	593.06	210.24	0 1 10
594.86	211.15	0 6 5	588.11	211.10	0 6 1	592.04	210.23	0 2 5
595.11	211.15	0 6 8	587.40	211.10	0 6 4	592.29	210.23	0 2 8
604.11	211.21	0 6 11	594.56	211.15	0 6 1 0	601.18	210.28	0 2 11
592.04	210.23	0 9 5	590.86	210.23	0 9 1	590.15	210.22	0 4 7
592.29	210.23	0 9	590.15	210.22	0 9 4	593.78	210.24	0 4 10
601.18	210.28	0 9 11	597.41	210.26	0 9 10	592.54	210.24	0 5 8
209.63	85.83	1 3 5	210.62	83.05	1 3 6	605.74	210.30	0 5 11
208.78	331.84	1 4 3	591.45	210.04	1 2 5	597.41	210.26	0 7 10
591.82	210.16	1 4 5	591.57	210.00	1 2 8	619.41	210.37	0 8 11
591.94	210.12	1 4 8	595.95	208.61	1 2 11	248.69	-23.81	1 3 10
596.33	208.73	1 4 11	595.94	208.61	1 3 6	252.06	-33.26	1 3 11
591.45	210.04	1 7 5	596.10	208.56	1 3 9	596.86	208.56	1 4 3
591.57	210.00	1 7 8	246.91	319.66	1 5 6	597.02	208.51	1 4 6
595.95	208.61	1 7 11	591.70	209.96	1 5 8	596.10	208.56	1 7 3
592.55	210.40	1 10 5	598.15	207.91	1 5 11	596.26	208.51	1 76
592.68	210.36	1 10 8	596.26	208.51	1 6 9	248.72	-10.53	1 8 10
597.08	208.97	1 10 11	604.58	205.86	1 8 11	251.89	-19.13	1 8 11
727.18	1822.53	3 5 6	591.09	209.92	2 5 4	598.36	208.56	1 10 3
1101.77	2899.70	3 5 10	594.71	211.10	2 5 10	598.52	208.51	1 10 6
209.98	335.29	3 6 8	542.18	1243.63	2 5 6	249.64	449.34	2 3 10
3930.00	11032.44	3 6 10	688.08	1647.23	2 5 10	246.38	439.96	2 3 11
188.45	273.37	3 6 11	1042.92	2628.89	2 6 10	586.90	208.56	2 5 3
249.64	449.34	3 9 10	249.61	434.25	2 9 10	586.75	208.51	2 5 6
246.38	439.96	3 9 11	246.55	425.79	2 9 11	249.61	434.25	2 7 10
248.74	446.76	3 10 11	248.82	432.06	2 10 11	246.55	425.79	2 7 11
587.05	208.61	5	3 6 591.21	209.88	2 8 4	587.14	208.56	2 8 3
586.90	208.56	5	3 9 594.84	211.06	2 8 10	586.99	208.51	2 8 6
247.39	98.12	5 4 6	595.59	208.49	2 11 4	595.80	208.56	2 11 3
591.09	209.92	5 4 7	599.29	209.67	2 11 10	595.65	208.51	2 11 6
592.91	210.52	5 4 10	479.03	42646.15	3 4 6	280.19	15696.88	3 4 10
586.75	208.51	5 6 9	200.76	4932.29	3 4 8	197.60	4504.18	3 4 11
594.71	211.10	5 7 10	197.44	-2006.71	3 5 6	238.89	-4918.27	3 5 10

Table 3-5: The vanishing points generated for texels # 0,7 and 9 for figure 3-5

constraints at this level of abstraction.

shape-from-parallel-lines continued								
		Texel # 2			Texel # 7			Texel # 9
V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used	V P Row	V P Col	Texels Used
542.18	1243.63	7 5 6	222.32	-3754.46	3 5 8	696.63	-37073.97	3 5 11
688.08	1647.23	7 5 10	163.95	345.79	3 5 10	249.60	429.49	3 7 10
1042.92	2628.89	7 6 10	595.21	208.61	3 6 4	246.60	421.57	3 7 11
249.61	434.25	7 9 10	602.67	208.61	3 6 10	248.73	-6.47	3 8 10
246.55	425.79	7 9 11	456.79	976.42	3 5 6	251.84	-14.54	3 8 11
248.82	432.06	7 10 11	106.75	52.39	3 5 8	595.37	208.56	3 4 7
587.30	208.61	8 3 6	532.62	1176.58	3 5 10	599.10	208.56	3 4 10
587.14	208.56	8 3 9	107.00	53.05	3 5 11	587.38	208.56	3 5 8
247.40	98.10	8 4 6	654.64	1498.69	3 6 10	600.22	208.56	3 5 11
591.21	209.88	8 4 7	107.71	54.92	3 8 11	602.84	208.56	3 7 10
593.03	210.48	8 4 10	249.60	429.49	3 9 10	613.47	208.56	3 8 11
586.99	208.51	8 6 9	246.60	421.57	3 9 11	218.94	-14876.42	4 5 10
594.84	211.06	8 7 10	248.84	427.48	3 10 11	134.82	9734.84	4 5 11
595.96	208.61	11 3 6	105.79	364.62	3 8 4	248.93	93.41	4 6 10
595.80	208.56	11 3 9	106.08	363.88	3 8 10	595.52	208.51	4 7 6
595.59	208.49	11 4 7	595.37	208.56	3 9 4	599.26	208.51	4 10 6
597.44	209.08	11 4 10	602.84	208.56	3 9 10	247.79	322.41	5 6 11
595.65	208.51	11 6 9	182.86	-4319.43	4 5 6	587.23	208.51	5 8 6
599.29	209.67	11 7 10	332.11	-47989.13	4 5 8	600.06	208.51	5 11 6
			246.91	319.64	4 5 6	266.73	8981.15	6 7 10
			591.34	209.85	4 5 8	224.41	2270.31	6 7 11
			597.78	207.79	4 5 11	243.47	-2628.65	6 8 10
			595.52	208.51	4 6 9	-397.00	54045.36	6 8 11
			604.20	205.75	4 8 11	603.00	208.51	6 7 10
			594.96	211.02	5 8 10	613.29	208.51	6 8 11
			601.53	208.97	5 11 10	232.78	-7967.88	7 8 10
			266.73	8981.15	6 9 10	202.92	4060.77	7 8 11
			224.41	2270.31	6 9 11			
			232.91	3618.21	6 10 11			
			603.00	208.51	6 9 10			
			162.64	346.20	6 11 10			
			232.78	-7967.88	8 9 10			
			202.92	4060.77	8 9 11			
			175.44	15131.98	8 10 11			
			608.08	206.92	10 8 11			

Figure 3-5: continued

In this chapter three levels of abstractions were discussed: the pixel level,

the surface patch level, and the surface level. Of these three, the surface patch level was chosen since it limits the data explosion problem that occurs at the pixel level, while allowing for the derivation of surface parameters for complex planar and non-planar surfaces.

The "texel patch" consisting of one or more texels has been defined. The specifics of the texel patch are determined by the texel patch finding algorithm and can map to one of the measurement-units described in the previous chapter. The three major classes of texel patch generators corresponding to the three types of measurement-units were defined. A specific texel patch generator that finds blobs by applying a simple bin thresholding scheme to generate threshold ranges, and defines a texel patch as a group of four-connected pixels, was described.

A new data structure, the *augmented texel*, was defined which combines multiple constraints on surface orientation in a compact notation for a single surface patch. Each augmented texel is comprised of a texel patch, orientation constraints for the texel patch, and a correctness weighting for each orientation constraint. The correctness weightings are comprised of an intra-cue weighting and an inter-cue normalization factor. The intra-cue weighting is defined based on the underlying assumptions of the shape-from-texture method. The inter-cue normalization factor, which is defined in the next chapter, assures that the number of constraints generated by each shape-from-texture module is not a factor in deciding what the correct orientation is; rather, the orientation that is most prominent overall.

The orientation constraints, stored in the form of vanishing points, are generated by the system's several independent shape-from-texture modules. These texture components, which run autonomously and may run in parallel, derive constraints by any of the currently existing shape-from-texture approaches. Five shape-from-texture cues, shape-from-uniform-texel-spacing, shape-from-uniform-size, shape-from-relative-eccentricity, shape-from-parallel-

lines, and shape-from-eccentricity have been defined or redefined to generate weighted orientation constraints based on the augmented texel paradigm.

4. Orientation Constraint Integration

The previous chapter described the first stage of a multiple shape-from-texture system. During this first stage (see figure 3-4) *texel patches*, each of which contain one or more texels, are found and passed to the system's many shape-from-texture cues. Each of these cues will generate zero or more weighted orientation constraints for each of the texel patches. The individual texel patches and their weighted orientation constraints are stored in a new data structure, the *augmented texel.*

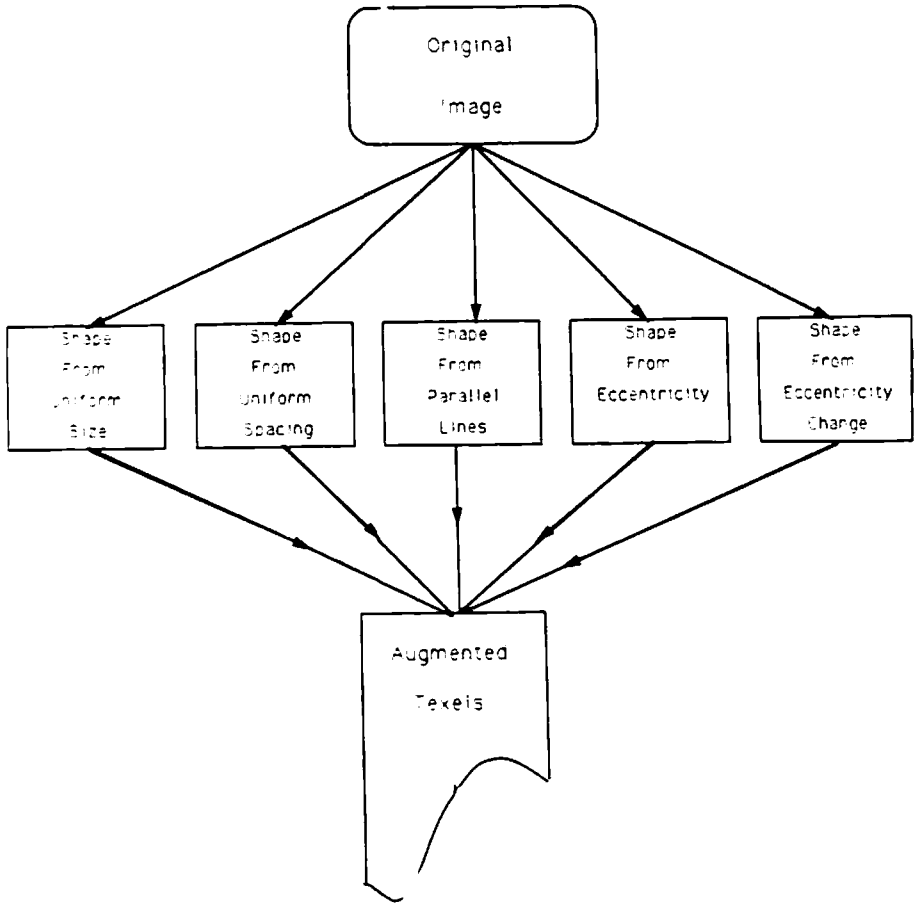


Figure 3-4: Constraint generation phase

Next, a method must be derived that combines each augmented texel's orientation constraints into a single orientation. This combination or *fusion* of constraints is comprised of two components:

- *The fusion methodology* that takes each augmented texel's

individually weighted orientation constraints and generates a single "most likely" orientation for the augmented texel's texel patch.

- *A constraint normalization* factor that is multiplied by the intra-cue weighting component of each orientation constraint creating a globally consistent weighting. This factor is necessary to assure that cues are weighted equivalently even though they potentially generate different orders of constraints. (Given a texel *T* some cues utilize only the single texel *T* thus generating only one constraint. Other cues utilize all groupings of four texels that include texel *T*, generating $O(n^3)$ constraints.)

The first half of this chapter (sections 4.1 to 4.2.5) defines what are the appropriate criteria for selecting a fusion approach, and then chooses a fusion approach that fulfills these criteria. The remainder of the chapter (sections 4.3 to 4.3.5) derives an inter-cue normalization factor, whose performance is compared to other potential normalizations on test imagery. Although the normalization factor is described after the fusion method, in the operational system the normalization can occur either prior to or simultaneously with the fusion step.

4.1 Choosing a Fusion Approach

The major difficulty in computing the "most likely" orientation is that the constraints are errorful, inconsistent, and potentially incorrect. Part of this problem is solved by the combination of the intra-cue correctness weighting (defined in section 3.2.6) and the inter-cue normalization factor (that will be discussed in section 4.3) which together define a heuristic likelihood that the constraint is correct.

Since no a priori knowledge exists as to which shape-from-texture cues are applicable, the constraint weightings are approximate and the way in which the constraints are fused is important. The following are three major design criteria for fusion, based on the errorful properties of real camera acquired textured surfaces:

1. Some of the constraints that are generated are incorrect due to the application of methods that are not valid for the textured surface.

2. Even among the valid orientation constraints, many of these constraints are errorful due to noise and digitization errors.
3. Even without noise, some shape-from-texture cues are themselves approximate. (E.g. shape-from-uniform-texel-size assumes paraperspective rather than true perspective.) Therefore, the fusion methodology must be able to consider approximate information.

The fusion methodology should not only be able to compensate for errorful data but should also be usable. Thus an additional criteria exists:

4. The method should be as computationally efficient as possible, and mappable onto a parallel architecture to assure real time operation.

After considering different fusion methods and their usefulness based on the four criteria above, a "graphical" solution[13] was chosen, specifically one which maps the space of all possible surface orientations onto a bin space. The number of bins specifies the accuracy to which the solution can be determined. The solution is found graphically by iterating though all of the constraints, and for each constraint incrementing the weighting of each bin for which the constraint is valid. The bin with the highest value is chosen as the result. This can be thought of as a Hough-like bin accumulation voting method applied to surface orientation.

In the specific problem of fusing orientation constraints, the solution space of all possible orientations is two dimensional. The two dimensional orientation space, which can be defined in many ways and can be discretized into a two dimensional array of bins. Several major orientation spaces exist and must be considered as the bin space. Two of the more common are the gradient space and the Gaussian sphere. The gradient space, originally introduced by Mackworth [Mackworth 73] utilizes a planar space of ordered pairs (p,q). If the normal of a plane is defined as:

---

[13]The primary alternative was a methodology that calculated orientations "symbolically" by using least squares approximations.

$$N = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1 \right)$$

then

$$p = \frac{\partial z}{\partial x} \quad q = \frac{\partial z}{\partial y}$$

This mapping of the surface normal in 3-space onto a point in the gradient space can be seen in figure 4-1. On the left hand side of the figure a planar surface (or surface patch) with its surface normal is shown in a 3-space coordinate system. On the right side is the gradient space equivalent representation of its orientation, a point on the cartesian (p,q) plane.

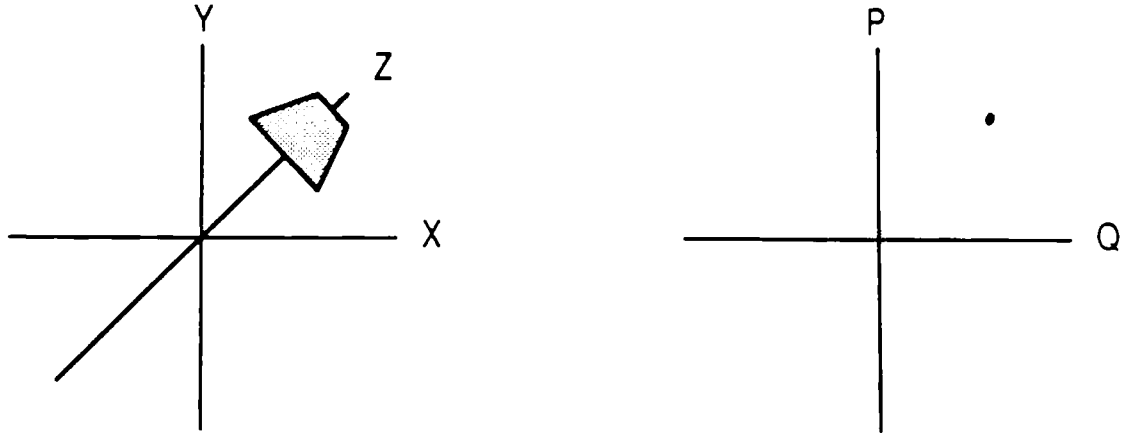


Figure 4-1: A surface in 3 space and its gradient space orientation

Orientation constraints map onto the (p,q) space simply. A vanishing point in image space becomes a line in (p,q) space; a vanishing line, which uniquely defines a surface orientation, maps onto a single point in (p,q) space.

The gradient space is infinite and continuous in two dimensions yet the bin space we wish to create is finite and discrete. Thus a non linear mapping of potential surface orientations to bins in the bins space would necessary.

The second orientation space, the Gaussian sphere, is closely related to the gradient space [Kender 80]. The Gaussian sphere is a unit sphere which maps all visible and invisible surface[14] orientations onto points on the sphere. Each point on the sphere defines a specific orientation (see figure 4-2) and any group of points specifies a class of orientations.

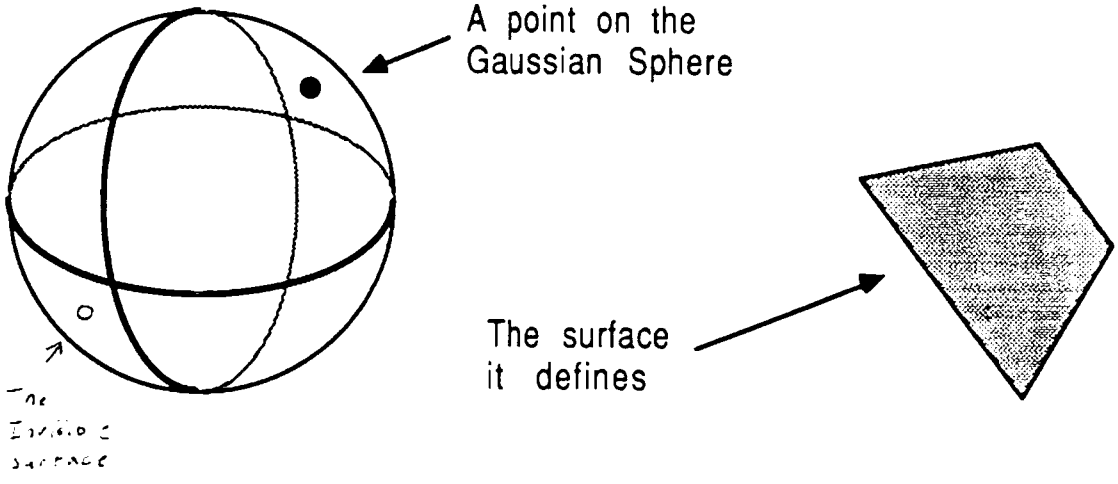


Figure 4-2: A point on the Gaussian Sphere
and the surface orientation it defines

A single orientation constraint (which can be defined by a vanishing point) circumscribes a great circle on the Gaussian Sphere ([Kender 80] page 61); two different constraints generate two great circles that overlap at two points uniquely defining the orientation of both of the sides of the surface patch.

The Gaussian sphere has a number of distinct advantages over the gradient space.

---

[14]The invisible surface is defined as the side of the surface that is on the opposite side of the surface from the viewer.

1. The gradient space is an infinite 2-dimensional space. Points at infinity, in any direction, define surfaces which are parallel to the Z axis.
2. Thus, highly oriented surfaces are easier to represent in the Gaussian sphere than in the gradient space. This can be seen by considering the picture plane as being placed on the same coordinate system with the Gaussian sphere and the gradient space. Then the Z axis maps into the viewers line of sight. Surfaces parallel to the Z axis in the Gaussian Sphere representation are simply points on the sphere where Z = 0.
3. The Gaussian sphere can be discretized by hierarchical tesselation.

4.2 The Tesselated Gaussian Sphere

The continuous Gaussian sphere can be approximated within the system by a finite tesselated sphere (see figure 4-3). This is acceptable since the orientation constraints are generated by the shape-from-texture cues which are approximate in nature. The shape-from-texture cues are approximate due to two major factors:

1. The original image that the texture information is derived from, is approximate. The image is a discretization of the real world. In the real world texture properties, such as size or spacing, can exactly be calculated. Given a digitized image, texture properties can only be determined within the precision of the digitization.
2. The cues themselves are based on simplified distortion models (e.g. orthographic, paraperspective, etc.) that attempt to simulate perspective distortion. The simplification is necessary due to the inherent mathematical complexity of perspective distortion.

To obtain as exact a solution as possible, the number of faces comprising the tesselated Gaussian sphere will be large (see figure 4-8). For each orientation constraint, each face would have to be examined to determine if the constraints are valid for the face. Given the large number of faces and a potentially large number of constraints this can become a very computationally expensive process. To allow the accuracy to be varied, as well as to decrease the complexity, a hierarchical tesselation scheme is used.

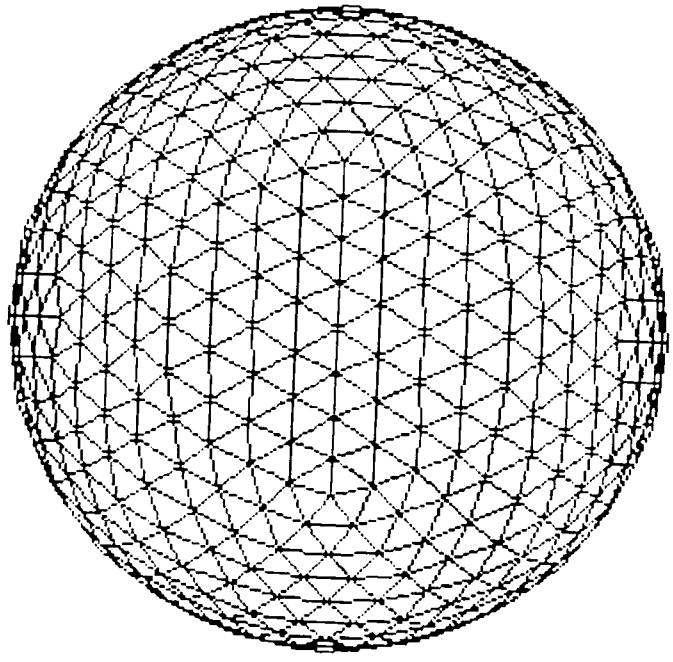


Figure 4-3: One possible tesselation of the Gaussian Sphere

This scheme is based on triangular shaped faces called trixels (See figure 4-3) [Fekete and Davis 84; Korn and Dyer 86]. The top level of the hierarchy is the twenty face icosahedron (see figure 4-4). At each level, other than the lowest level of the hierarchy, each trixel has four children which more closely approximate the curvature of the spherical surface than their parent. This hierarchical methodology allows the user to specify the accuracy to which the orientation can be calculated by defining the number of levels of tesselation that are created.

4.2.1 Generating the Tesselated Gaussian Sphere

A hierarchical tesselated Gaussian Sphere is created as follows. First, a 3-dimensional coordinate system is constructed in which the center of the Gaussian sphere is located at the center of the coordinate system. On an infinitely tesselated sphere all points on the sphere are a unit distance from the center.

The first level of the hierarchy is an icosahedron which is constructed by calculating the position of each of the 12 vertices that define its 20 faces. The

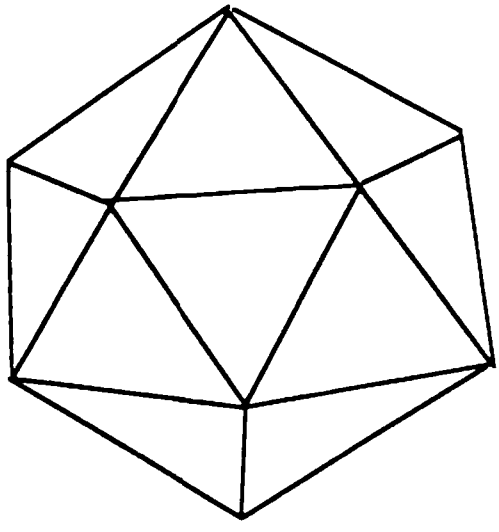


Figure 4-4: The 20 face icosahedron

coordinates of each of these vertices is calculated from the golden ratio *G* [Ballard and Brown 82] where:

$$G = \frac{1 + \sqrt{5}}{2}$$

and

$$A = \frac{\sqrt{G}}{5^{1/4}} \qquad B = \frac{1}{5^{1/4}\sqrt{G}}$$

The 12 possible vertices are placed at

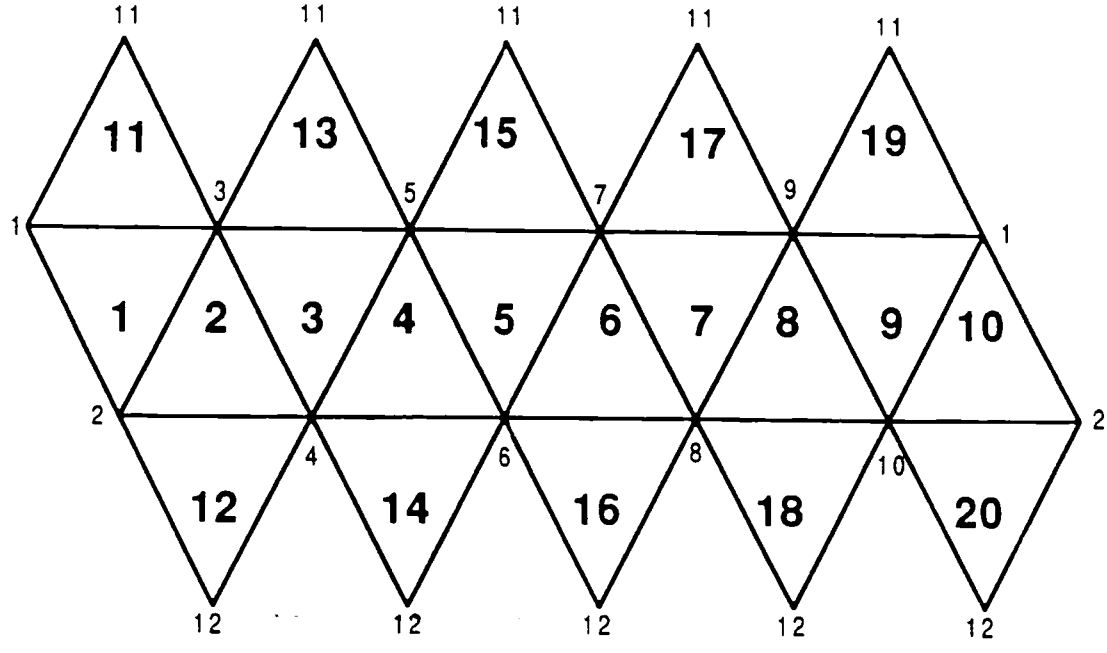
Vertex #	x coordinate	y coordinate	z coordinate
1	0	A	B
2	0	A	-B
3	A	B	0
4	B	0	-A
5	A	-B	0
6	0	-A	-B
7	0	-A	B
8	-A	-B	0
9	-B	0	-A
10	-A	B	0
11	B	0	A
12	-B	0	-A

Once the vertices have been created the twenty faces are assigned to the vertices. The icosahedron can be better visualized by picturing it as flattened out (See figures 4-5 and 4-6 for one possible final vertex assignment). It should be noted that the numbering of the faces, as well as the numbering of the vertices can be arbitrary; the only requirement is that faces should be numbered in a way that allows one to calculate which faces are neighboring.

Once the top level faces have been created, an iterative algorithm is applied to create four children out of each parent face (see figure 4-7). Each of the parent's edges are bisected creating three new vertices. The vertices of the parent lie on the surface of the sphere, but these new vertices lie within the sphere. The distance d from each new vertex V to the center of the sphere is calculated:

$$d = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

Since the sphere is a unit sphere, each point on the sphere is 1 unit distant from the center of the sphere. Each vertex is therefore "pushed out" onto the



Face #'s are in bold type and vertex #'s are in light type

Figure 4-5: The Icosahedron flattened out with vertex numbering

Face #	vertex # 1	vertex # 2	vertex # 3	Face #	vertex # 1	vertex # 2	vertex # 3
1	1	2	3	11	1	11	3
2	2	3	4	12	2	12	4
3	3	4	5	13	3	11	5
4	4	5	6	14	4	12	6
5	5	6	7	15	5	11	7
6	6	7	8	16	6	12	8
7	7	8	9	17	7	11	9
8	8	9	10	18	8	12	10
9	9	10	1	19	9	11	1
10	10	1	2	20	10	12	2

Figure 4-6: The faces of the icosahedron

unit sphere by dividing each of its coordinates by *d*. In this way each child is

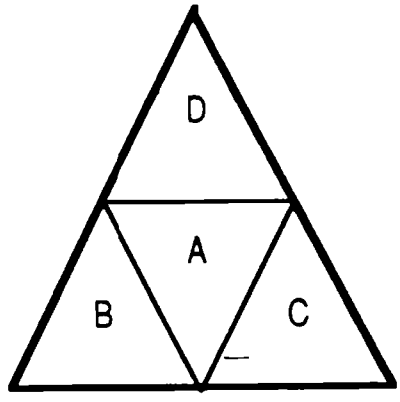


Figure 4-7: Tesselating a trixel into four children

generated.

This tesselation scheme is simple to implement, yet it has one major disadvantage. This tesselation scheme does not create trixels of equivalent area. Therefore each trixel defines an orientation to within a different accuracy. Furthermore, trixels of larger area will have a larger probability of having more constraints fall on them. Conceptually, a trixel may be chosen over its neighbors simply because it covers more area. By using a higher degree of refinement this problem is somewhat alleviated. However, experimental results seem to show that the problem is not very important.

The final step in tesselating the sphere is to set up the neighbor relationships among the children. This information is used to simplify the smearing phase (described in section 4.2.4) of the orientation calculations. If the children are numbered as in figure 4-7, then child *A*'s neighbors are *B,C,D* while one of *B,C,D*'s neighbors is *A*. The remaining two neighbor relationships of children *B,C,D* are generated from the neighbors of the parent. For example child *B*'s left neighbor is its parent's left neighbor's child *D* and *B*'s lower neighbor is *B*'s parent's lower neighbor's child *B*.

Since the hierarchically tesselated Gaussian Sphere is used as a bin-based accumulator, each trixel will contain weighting information.

4.2.1.1 Deriving how tesselated a sphere to use

Once the mechanism has been created to generate the tesselated Gaussian Sphere, the next step is to determine how refined a tesselation is necessary. Each additional level of tesselation creates trixels that are more exact than the previous level as shown in figure 4-8. Specifically the error is halved with each additional level of tesselation. If the constraints were exact, rather than approximations, one would want to use as many levels, are possible within the computing time. However, the constraints are approximate and the accuracy of the solution can be only be as exact as the constraints.

In order to determine the accuracy of the solution, a large number of factors must be considered including: the discreteness of the imaging technology, whether the algorithm used to generate each shape-from cue is approximate, and the specific computational model used.

However, these factors are dominated by the fact that many shape-from-texture cues have errors greater 1 degree and that digitization errors can cause at least a 1 degree error. In fact, some researchers consider a 3 to 5 degree error bound sufficient (e.g. [Brown 86]). Therefore only five teselation levels are required. Each additional level increases the number of faces by four while only decreasing the error by 1/2.

It is important to note that if additional accuracy is required a selective, dynamic subdivision can be utilized. Under this scenario the chosen trixel and its neighbors could be tesselated and the constraints re-applied to the additional trixels of higher accuracy only when necessary. If the number of constraints is smaller than the number of trixels at the sixth level (20480) then a time savings can be gained.

This approach works when the system is supplied either orientation

tesselation level	# of faces at this level	Total # of faces	accuracy of the orientation in degrees	
1	20	20	p = ±17.27	q = ±17.27
2	80	100	p = ±9.14	q = ±8.33
3	320	420	p = ±4.62	q = ±4.36
4	1280	1700	p = ±2.38	q = ±2.30
5	5120	6820	p = ±1.21	q = ±1.19
6	20480	27300	p = ±0.61	q = ±0.61

Figure 4-8: The tesselation levels and their accuracy

constraints (i.e. great circles) or potential orientations (i.e. points on the sphere). In the next two subsections the methodology necessary to fuse both kinds of information are discussed.

4.2.2 Applying Orientation Constraints to the Gaussian Sphere

The system generates the "most likely" orientation for each texel patch by accumulating the evidence from all the orientation constraints for the patch. For each constraint, evidence is accumulated by visiting the twenty top level trixels, determining whether the great circle falls on the trixel, and if the result is positive, visits the children. At each lowest level trixel through which the great circle travels, the likelihood value of the trixel is incremented by the constraint's weight. The hierarchical nature of this approach limits the number of trixels that need to be visited.

An orientation constraint, in terms of a vanishing point (V_x, V_y), defines a surface $((V_x \times a) + (V_y \times b) + c = 0)$ in the Gaussian Sphere's coordinate system. To determine if a specific trixel is valid for the orientation constraint, one must simply check to see if the vanishing point's plane travels through the trixel. The trixel is a planar surface of known position and known boundaries (defined by the

three corners) while the vanishing point's plane is an infinite plane. The question then is one of calculating if the infinite plane travels through the finite surface. One simple way of calculating this is from the standard equation for measuring the distance *D* from a point *a,b,c* to a surface

$$D = \frac{(V_x \times a) + (V_y \times b) + c}{\sqrt{V_x^2 + V_y^2 + 1}}$$

To determine if the plane is located somewhere within the triangular area defined by the three corners of the trixel one need not calculate the actual distance from each of the three points to the surface, but rather to calculate the direction which the surface lies in relationship to each of the three corners. The sign of the distance is simply

$$S = (V_x \times a) + (V_y \times b) + c$$

By calculating *S* for each of the three corners and determining if *S* is positive for at least one of the corners and negative for at least one of the corners then one can determine if the surface lies within the trixel. Thus the constraint is valid for the range of surface orientations defined by the trixel.

The only remaining question is to decide whether to count a surface that falls exactly on one of the corners ($S = 0.0$ for one of the corners) as falling on the trixel itself. If the constraint is defined as falling on the trixel then it will be included in the weighting of all three trixels that meet at the corner. This can only occur in a tiny fraction of the constraints, therefore the choice is not very important. In fact the implementation that is used in the test examples (see chapter 7) considers constraints falling exactly on the corner as falling on the trixel itself.

The shape-from-texture fusion paradigm can be best understood by a simplifed example. Let us consider texel number 2 of the circles image of figure 3-5 which contains 71 orientation constraints. To simplify the example let us only consider the 11 constraints which were generated by shape-from-uniform-texel-size (see figure 3-2). Each of these constraints increments the weights of a

subset of the trixels that comprise the tesselated Gaussian sphere. Figure 4-9 shows which grouping of trixels will be incremented for each of the constraints. Only the visible orientations are shown, since the invisible faces are a mirror image of the visible faces (i.e., their Necker reversal).

4.2.3 Applying Orientations

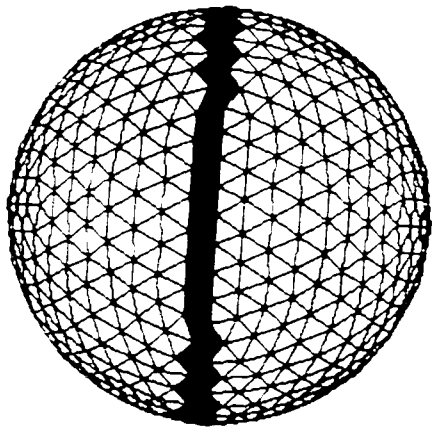
Some shape-from-texture cues generate potential orientations rather than orientation constraints. This variation is a modification of the orientation constraint application method described in the previous subsection. Rather than checking to see if a constraint falls in the trixel the system checks to see if the orientation is within the trixel. The validation is performed by checking to see that for each of the three edges of the trixel the orientation is on the same side of the edge as the center of the trixel.

4.2.4 Smearing Constraints

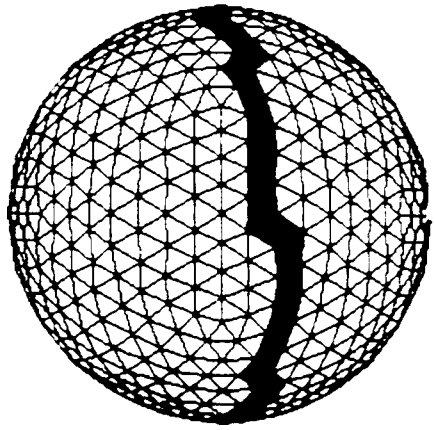
The multiple shape-from-texture system must be resilient to incorrect constraints, approximate constraints, and constraint perturbations Weight smearing has been chosen as the best method for dealing with these problems.

Once all of the constraints for a texel patch have been considered, the likelihood values at the lowest level trixels are smeared. Currently, this is done heuristically: the smeared value of each leaf is equal to its accumulated value plus 1/2 the value of its neighbors plus 1/4 the value of all its neighbor's neighbors that are not a neighbor of the leaf. This is a rough approximation to a gaussian blur. The "most likely" orientation is defined to be the trixel with the largest smeared value.

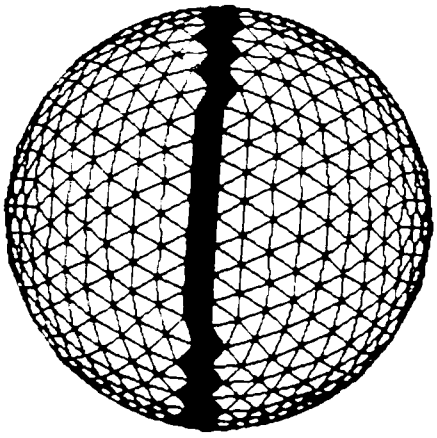
Since different cues have different error statistics, a more exact solution could be obtained by modeling the error of the constraints generated by the different cues. The system could accumulate the weighting value generated by each of the cues separately, and the cue specific smearing method would be applied. The resulting smeared weight would be added to the accumulated total



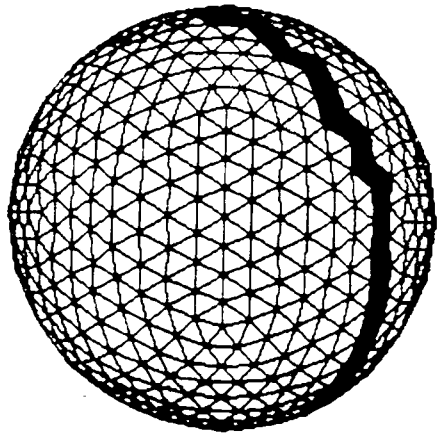
p = 452.786072 q = -19310.093750



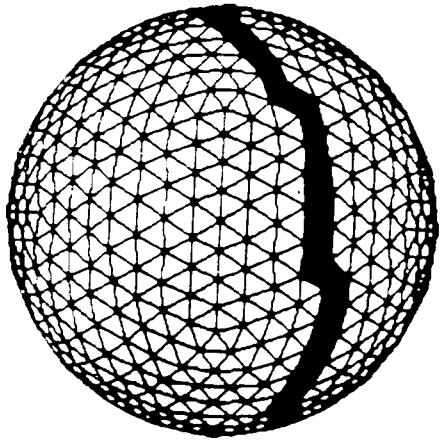
p = 620.500732 q = 219.492386



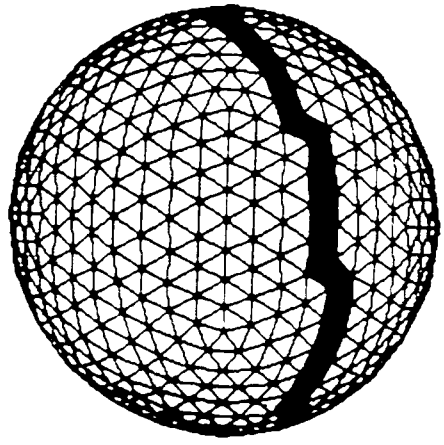
p = 321.267700 q = -49610.113281



p = 622.635437 q = 848.681030



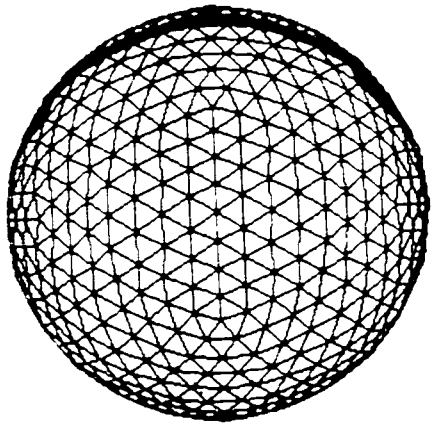
p = 605.250977 q = 1471.918823



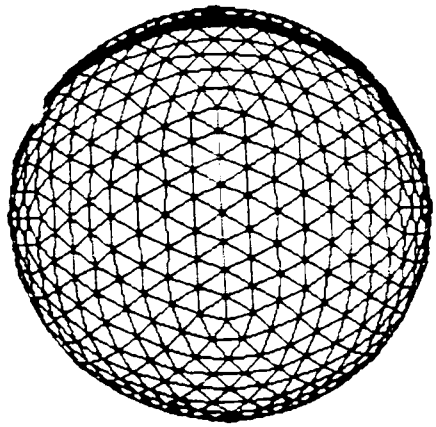
p = 620.717285 q = 1460.889771

*To simplify the display the Gaussian sphere contains only 5120 texels. Only the leaves are shown.

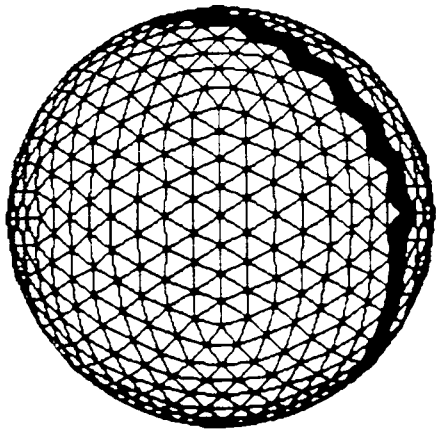
Figure 4-9: 6 example constraints applied to the gaussian sphere



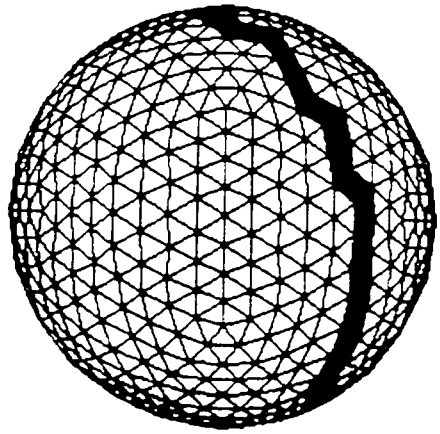
p = 620.481018 q = 2757.755371



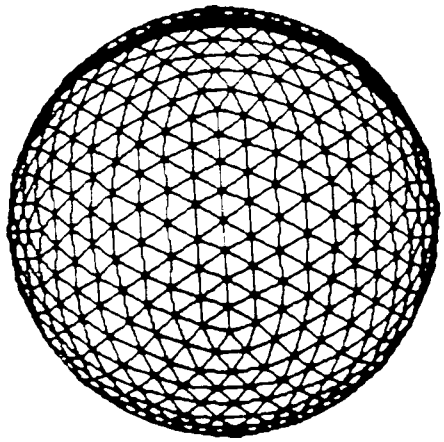
p = 626.741577 q = 221.436066



p = 637.133545 q = 646.014343



p = 632.649780 q = 1059.339478



p = 646.557739 q = 224.767441

*To simplify the display the Gaussian sphere contains only 5120 texels.

Only the leaves are shown.

Figure 4-10: 5 example constraints applied to the Gaussian sphere weight for the trixel.

Experimentally however, the simple smearing method is able to aid in determining the correct solution. The smearing of each trixel's weighting to its neighbors and its neighbors' neighbors creates a higher weighting for the correct trixel than the other trixels, as shown by the darkness of one trixel in relationship to other trixels.

4.2.5 Iterative Constraint Propagation

The multiple shape-from-textures fusion method, as described, does not assure that under all circumstances a single "most likely" orientation will be derived. The system is supplimented with an iterative constraint propagation algorithm.

During each iteration the orientation of a single texel is solved as follows: The system computes the "most likely" orientation for any unresolved texel patches, and then marks as resolved the texel patch whose "most likely orientation" is greater than its second "most likely orientation" by the largest percentage. The system then re-analyzes each of the remaining texel patches. Re-analysis is performed by considering all of the patch's constraints and removing any constraints that do not participate in the definition of the "most likely" orientation of the just resolved texel patch. This re-analysis does not assure a single "most likely" orientation either, but it does aid in simplifying and deriving a single "most likely" orientation for a large number of surface patches.

4.2.6 Mapping the gaussian sphere onto a parallel computer

The hierarchical tesselated Gaussian sphere can be considered as a variation on the standard quad-tree representation of areas. If a quad-tree parallel computer (where each node has four children) is used to implement the tesselated Gaussian sphere then a number of software implementation modifications will be necessary to map the sphere onto an existing quad-tree. Since the tesselated Gaussian sphere consists of 20 nodes at the top level then a number of nodes must be left unused in order to create a 20 node top level.

Either:

1. The first 3 levels of the quad-tree are used for information transmission and are not considered as part of the Gaussian Sphere. Since the fourth level consists of 64 nodes only 20 of the 64 are used and 44 nodes are left unused. At each level approximately 1/3 of all the nodes are unused.
2. Since only the visible surface orientation is required not all of the 20 faces need to be used. The third level, containing 16 faces, could therefore be used as the first level of the tesselated Gaussian sphere. Four of the faces of the icosahedron which are located on the invisible side of the sphere will not be generated.

The other major modification occurs because the leaf nodes are linked to their three neighbors even when the neighbors are of different parents. This link establishes local connectivity and is used in smearing values between texels. A software implementation can be created by simulating the communication between neighboring trixels by passing the data up the tree to parents, and then down to the neighbors. The major disadvantage of this system is a loss of time that occurs during the communication phase depending on the exact hardware configuration.

Figure 4-11: A simple four connected SIMD parallel processor

4.3 Generating an inter-cue normalization factor

Prior to defining exactly what and how an inter-cue normalization factor is created it is important to discuss why such a factor is needed. As stated in Chapter 2 all of the existing shape-from-texture cues make at least one major

assumption about the texture, which allows perspective distortion to be used to calculate orientation constraints. This general assumption must be valid for some grouping of texels on a surface in order for the shape-from-texture cue to generate correct orientation constraints.

If a priori knowledge of the texture(s) existed then the selection of which shape-from-texture cues to apply, as well as the integration of the orientation constraints so generated, would become trivial. Since no source of this information exists all of the shape-from-texture cues must be applied to the whole image. The separation of the incorrect constraints generated by non-applicable cues from the correct constraints must occur during the fusion process.

The separation is further complicated by the fact that each shape-from-texture cue generates a different number of constraints for the same image (for the maximum number of constraints that could be generated see figure 4-11), and the number of constraints generated depends on the specific texture characteristics of the image. Prior to defining exactly how the different cues are normalized, an example showing the diversity in the numbers of constraints generated is appropriate. Figure 4-13 shows a synthetic image containing 9 uniformly spaced and uniformly sized circles. For each texel each cue generates a different number of "valid" constraints. For example, consider the texel located in the lower left corner of figure 4-13. Since shape-from-eccentricity needs only one texel to generate each constraint, it will generate only 1 constraint.

Shape-from-uniform-spacing will generate one orientation constraint for each co-linear grouping of three texels (see figure 4-15). In the maximum case the cue would generate (n-1)(n-2) constraints. Since only three sets of texel patches are co-linear only three constraints are generated.

Both shape-from-uniform-size and shape-from-relative-eccentricity utilize all possible groupings of two texels to recover orientation constraints and therefore will generate 8 constraints on the orientation of each texel (see figure 4-16).

Lastly, shape-from-parallel-lines would generate a maximum of (n-1)(n-2)(n-3) constraints or 336 constraints. Given the 3 texel co-linearity requirement this is cut down to only 28 "valid" constraints (see figure 4-17).

Method	Texel Patches Used	Order
Eccentricity	1	1 = $O(1)$
Size	2	(n-1) = $O(n)$
Eccentricity change	2	(n-1) = $O(n)$
Spacing	3	(n-1)(n-2) = $O(n^2)$
Parellelism	4	(n-1)(n-2)(n-3) = $O(n^3)$

Figure 4-12: Summary of Shape-from-texture cues used for figure 3-5

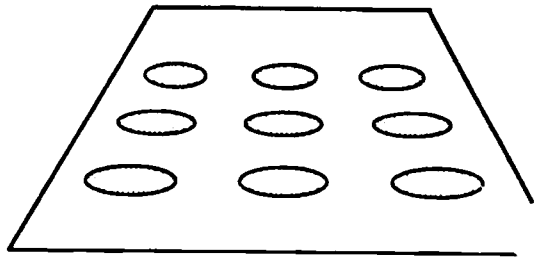


Figure 4-13: a surface covered with circles

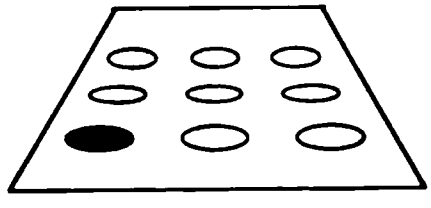


Figure 4-14: The lower left hand texel of a surface covered with circles

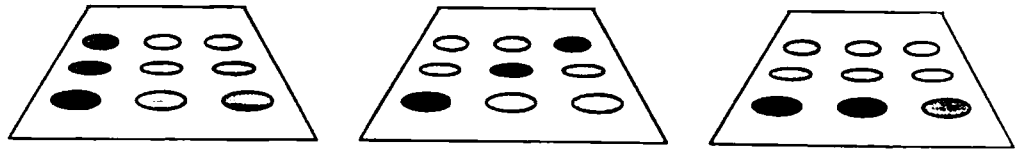


Figure 4-15: For the lower left corner texel shape-from-uniform-spacing will generate 3 constraints

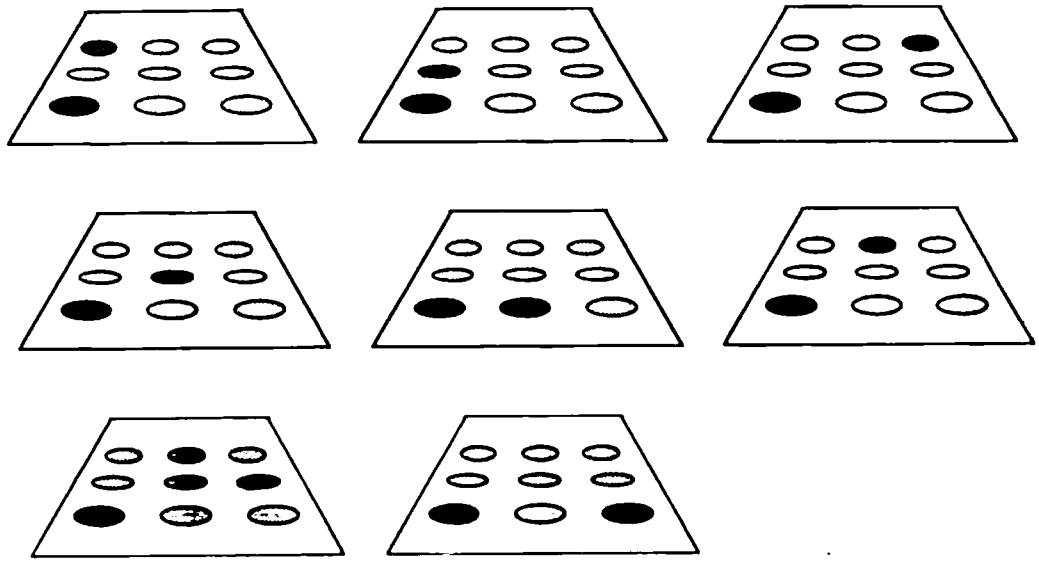


Figure 4-16: For the lower left corner texel shape-from-uniform-size will generate 8 constraints

The inter-cue normalization factor *N* is designed to assure that no single

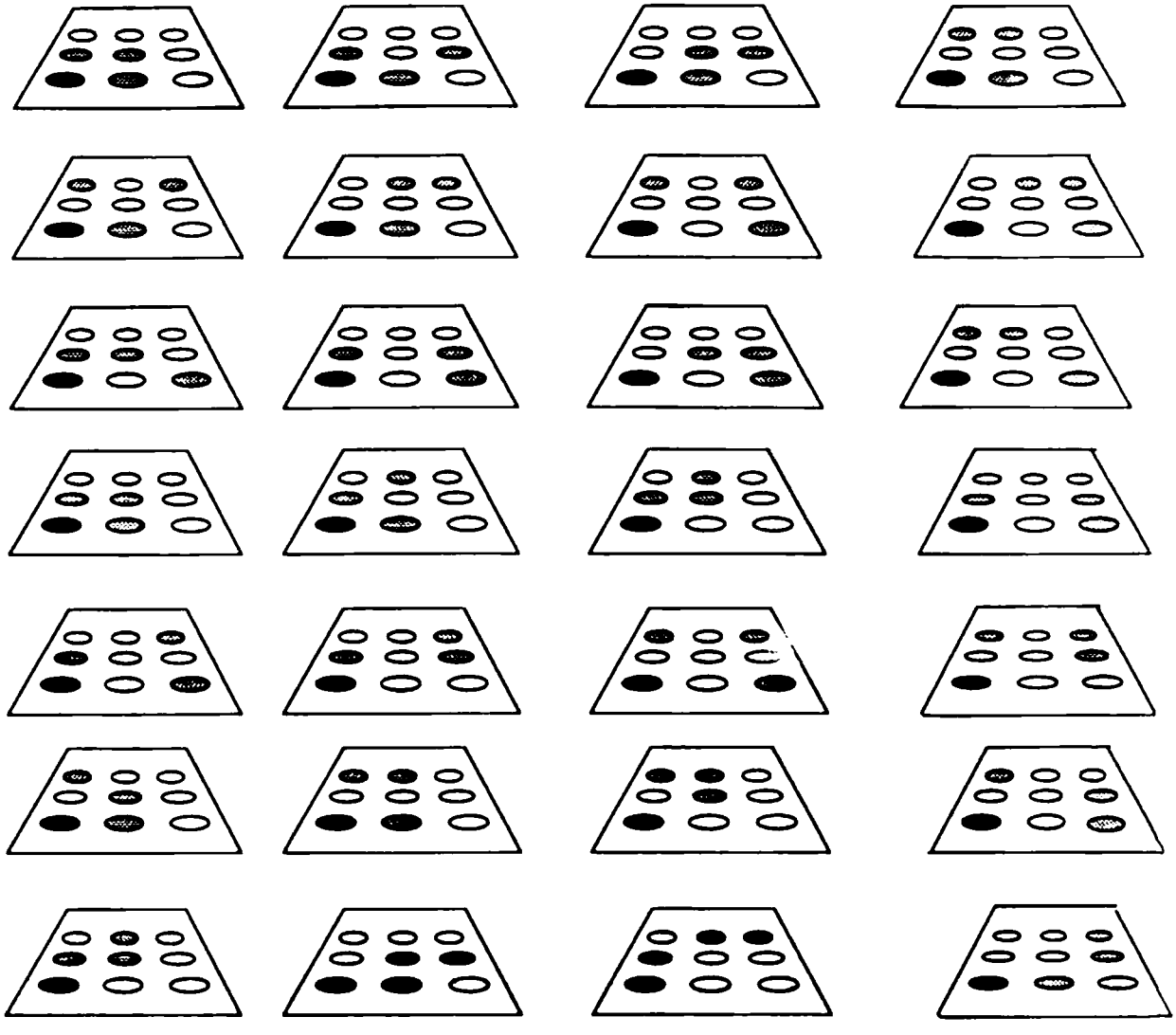


Figure 4-17: For the lower left corner texel shape-from-parallel-lines will generate 28 valid constraints

shape-from cue gains ascendancy over other shape-from cues simply because the cue generates more constraints (as does shape-from-parallel-texels). Furthermore, it aids in the separation of valid from non-valid cues. Four different weighting schemes have been considered and tested:

1. All of the constraints are equally weighted (i.e. one constraint, one

vote).

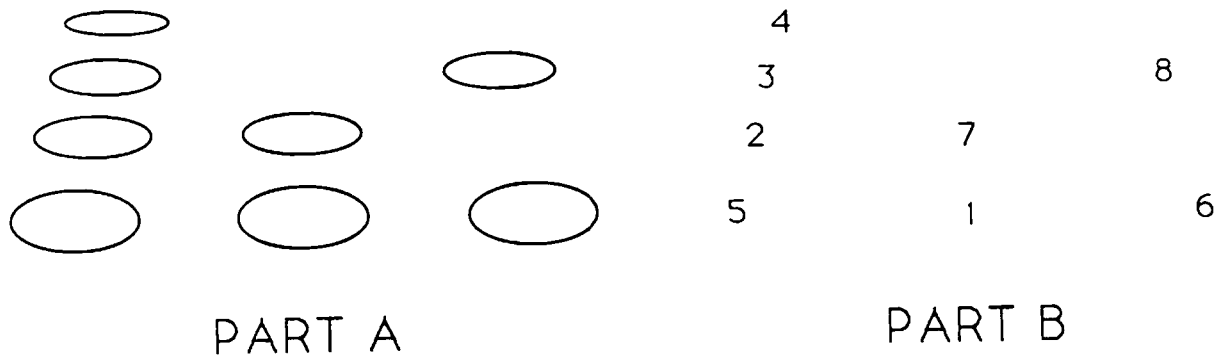
2. The weightings for all constraints generated by a single shape-from-texture cue are scaled such that the sum of the scaled weightings are equal to 1.0 (i.e. one cue, one vote).
3. The constraints generated by each cue are scaled in relationship to the maximum number of constraints that the cue could generate given the number of texels in the image.
4. The constraints generated by each cue are scaled in relationship to the "average" number of constraints that the cue could generate given the number of texels in the image.

Each of the next subsections will discuss one of the weighting schemes listed above, showing examples as to the advantages and disadvantages of the scheme.

4.3.1 All constraints are equally weighted

The first potential weighting scheme is to consider all of the constraints as having equal weight. In this scheme no knowledge is necessary as to which cues generate more constraints. Unfortunately, this scheme has a number of disadvantages.

Shape-from-texture cues when applied to images for which they are not valid generate conflicting noise-like orientation constraints. These noise-like constraints are 'randomly' distributed across the range of possible orientation constraints. One might assume that if all constraints are equally weighted, a single cue that generates conflicting noise-like constraints will have no effect on the orientation calculated. Unfortunately the constraints generated by non-valid cues are not truly random. For example let us consider the textured surface in figure 4-18a. The texels are not uniformly sized but they are co-linear and uniformly spaced. In order to simplify the example let us only consider the orientation generated for texel 1. (Texel numbering is shown in figure 4-18b). To further simplify the example consider that the intra-cue correctness factors generated for each of the constraints are all equal to 1.0.



The right hand side contains the numbering of the texels

Figure 4-18: A surface containing uniformly spaced texels

Shape-from-uniform-texel-size when applied to texels 1,2,3,4 will generate three orientation constraints for each of the texels (i.e. for texel 1 the constraints are from (1,2) (1,3) (1,4)). These vanishing points are co-linear with each other and with the four texels that were used to generate the constraints. This will mislead the system into deriving a very wrong surface orientation.

Shape-from-parallel-virtual-lines will not be able to generate any constraints. The only virtual line segments containing three or more texels are those connecting texels 1,2,3,4, texels 1,5,6, and texels 1,7,8. All of these segments intersect at a point, on texel 1, which is located on all three line segments. Therefore, by our heuristic definition of the cue, the intersection is ruled out as orientation constraint.

Shape-from-uniform-texel-spacing will generate a total of five constraints on the orientation of texel 1, one constraint for each of the three colinear texels. These groupings are (1,2,3) (1,2,4) (1,3,4) (5,1,6) (1,7,8).

All of these 5 constraints are consistent with each other and would, if the constraint is considered by itself, cause the correct orientation to be determined. When these constraints are considered in combination with the constraints from shape-from-uniform-texel-size the correct orientation is not chosen under this weighting scheme.

This occurs because the first three constraints of shape-from-uniform-texel-spacing (1,2,3) (1,2,4) (1,3,4) and the three constraints generated by shape-from-uniform-texel-size (1,2) (1,3) (1,4) all fall on the line connecting texels 1,2,3,4. These six co-linear constraints define a vanishing line with a weighting of 6. The five correct constraints generated by shape-from-uniform-texel-spacing define a vanishing line of weight 5. Therefore the incorrect orientation is chosen.

This problem would arise even if the intra-cue correctness factors are also considered. Since some cues can generate large numbers of incorrect constraints, while other cues generate fewer constraints which may be more accurate weighting each constraint equally was rejected.

4.3.2 Using weighting distributions

In the remaining weighting schemes, each cue is considered separately and an inter-cue normalization factor is generated. This normalization factor is used to equalize the effects that each of the cues has on the solution.

4.3.3 The sum of each cue's correctness weightings is equal to 1.0

In this weighting scheme each cue's constraints for a given texel patch are normalized such that the sum of the constraint weightings for each cue is equal to 1.0. The distribution of each cue's constraints in the space of all possible constraints thus simulates a probability distribution function. Each cue's distribution defines what the cue "believes" is the possibility is that each potential orientation is the correct orientation. In this scheme the intra-cue correctness factor is used as the initial weighting value of each constraint. The normalization factor is simply the sum of the values of all the lowest level trixels after the

application of the intra-cue weighted constraints. The trixels are then divided by this normalization factor.

Unfortunately, this weighting scheme fails when one of the cues generates only a single constraint. This occurs with certain shape-from-texture cues (e.g. shape-from-texel-eccentricity); it also occurs when a cue is applied to an image which contains few texels for which the cue is applicable (e.g. shape-from-uniform-texel-spacing applied to non-uniformly spaced texels).

This causes serious problems for the weighting scheme and would severely limit the usefulness of a system based on it. Obviously this is less serious than the flaws in the previous weighting scheme. Nonetheless, this weighting scheme should only be chosen if it no other scheme shows more resiliency to cumulative error.

4.3.4 Scaling by the Maximum Number of Constraints

The previous weighting schemes were unable to correctly deal with cues that generated fewer constraints then they were expected to. This occurs because certain shape-from cues are texel position sensitive (e.g. shape-from-uniform-texel-spacing requires three co-linear texels to generate each constraint). The next weighting scheme eleviates this problem by scaling each cue's constraints in relationship to the *maximum number* of constraints that the cue could generate given the number of texels in the image. This decreases the cumulative effects of cues generating erroneous constraints on surfaces they are not applicable to.

To better understand this weighting scheme let us consider the textured surface of figure 4-18 containing a total of eight texels. Given eight texels each of the four shape-from-texture cues will generate a different maximum number of constraints. Shape-from-eccentricity utilizes only one texel and thus will always generate one constraint for each texel. Shape-form-uniform-texel-size generates the same number of constraints (n-1 constraints if the image contains n texels)

no matter how the texels are positioned. The remaining two cues are texel position sensitive.

Shape-from-uniform-texel-spacing generates a maximum number of constraints if the texels are all co-linear. If the image contains n texels, then the maximum number of constraints generated for each texel is $\frac{(n-1)+(n-2)}{2}$. For the example image, shape-from-uniform-texel-spacing will generate a maximum of 21 constraints.

Shape-from-parallel-virtual-lines generates the maximum number of constraints if all the texels fall on two co-linear virtual lines. One of the virtuallines" contains the texel for which the constraints are being generated and only two additional texels. The second virtual line contains the remaining (n-3) texels. Shape-from-parallel-virtual-lines will generate for this maximum configuration

$$2 \times \frac{(n-4)^2 + (n-4)}{2}$$

that is $n^2 - 7n + 12$ constraints. Thus, for the textured surface in figure 4-18 the cue would generate 20 constraints.

Unfortunately, this scheme disproportionately decreases the effects of cues (such as shape-from-uniform-texel-spacing and shape-from-parallel-virtual-lines) which generate substantially less constraints, on the average, in relationship to cues that always generate the same number of constraints (such as shape-from-uniform-texel-size).

4.3.5 scaling by the "average" number of constraints

The last weighting scheme is based on the previous scheme. The difference is that this scheme utilizes an "average" number of constraints rather than the maximum number of constraints that the cue could generate given the number of texels in the image. This scheme has the advantage of the previous scheme: when the cue generates few constraints due to inapplicability the cue is effectively ignored. Furthermore, it does not disproportionately decrease the effects of cues that are sensitive to texel position.

The key issue is deciding exactly what is the "average" number of constraints that each cue generates. Obviously the "average" must be specific for each cue and a function of the number of texels in the image. The major factor in each cue's ability to generate constraints given a group of texels is the topology of the texels. An "average" must hypothesize a "general" layout, and the resulting numerical average is determined.

For some cues the "average" is equivalent to the maximum since the cue will always generate the same number of constraints no matter what the texel topology is. For example shape-from-uniform-texel-size will generate one constraint for each grouping of two texels while shape-from-eccentricity will always generate a single constraint for each texel.

The "average" becomes important for cues that are topology specific such as shape-from-uniform-texel-spacing and shape-from-parallel-virtual-lines. In each of these cues texels must be aligned in order for the constraints to be generated at all. Therefore, each of the two cue's "average" layout topology must consist of co-linear texels. After considering a number of different topologies and the number of constraints that are generated for each topology a simple two dimensional grid of uniformly spaced texels has been found to be both the simplest to implement and the closest to an "average" in a large variety of experimental runs.

Specifically, since the cues are based on measured changes in textural uniformity, the grid is an obvious choise since it is the most uniform 2-dimensional structure. Thus it fulfills the position based uniformity requirements of all of the shape-from-texture cues under consideration as follows:

1. Shape-from-uniform-texel-spacing requires that the texels are uniformly spaced. This is correct in a grid structure.
2. Shape-from-uniform-texel-size requires no position specific requirements , thus the grid creates no difficulties.
3. Shape-from-eccentricity and shape-from-eccentricity change also require no position specific requirements.
4. Shape-from-parallel-virtual-lines requires that there exist at least two groupings of two or more co-linear texels. This requirement is valid horizontally, vertically, and diagonally Thus a grid would allow the cue to generate a very large number of constraints.

4.4 Summary

In this chapter the problem of how to integrate conflicting and corroborating orientation constraints per surface patch has been confronted. The problem was split into two basic components: how to integrate the constraints, and how to weight each cue's constraints such that a well-founded result is chosen.

In sections 4.1 and 4.2 a criteria was given for choosing the best fusion method and then a method was chosen based on the criteria. The fusion method is based on a hierarchical tesselated Gaussian sphere. The tesselation allows for a simple mapping of bin-based fusion onto the Gaussian sphere. The hierarchial nature of this implementation speeds up the application of orientation constraints or full orientations.

In section 4.3 a number of different normalization factors were considered and an "average" number of constraints were chosen as the best. The advantage of a cue specific "average" function is that it can be defined independently for each cue.

Each of the current cues that are being fused generates either a fixed number of constraints based on the number of texels, or a variable number of constraints based on the number of texels and the topology of the texels' layout. For these cues the "average" is a cue specific function that utilizes an "average" layout topology. Each cue's "average" layout topology is instantiated with the number of texels in the average and determines an average number of constraints. This average is the inter-cue normalization factor.

5. Experimental results

In this chapter 5 different camera acquired images will used to test the multiple shape-from-texture system. Each of the images is designed to test a different aspect of the fusion process.

5.1 Single Textured/Single Surface Images

Initial testing of the multiple shape-from-texture cues system is images containing a single surface with a uniform texture. In the previous chapters a example image, containing 12 circles, has already been given (see figure 3-5). In this section 4 additional images of uniformly textured surfaces are given. The images differ in the cues that are applicable to the image, and in the amount of noise.

5.1.1 Computer Terminal Keyboard

The next image, figure 5-1 , shows a camera-acquired image of a computer terminal keyboard. The key tops are chosen by the system as texels due to the grey level disparity between them and the majority of the pixels in the image. The key top texels, shown in figure 5-2, are approximately uniformly sized. They are also uniformly spaced horizontally, and approximately uniformly spaced vertically. Unfortunately, they are only co-linear in the the horizontal direction, therefore the only constraints generated by shape-from-uniform-texel-spacing are in the horizontal direction.

The key top texels are poorly defined in the image due to digitization and the fact that the camera was not perfectly focused. Furthermore, many of the key tops are inscribed with letters and have shadows cast upon them. Therefore, the threshold based blob finding algorithm has difficulty in correctly recovering texels, as shown in figure 5-2.

None of the individual shape-from-texture cues by themselves derive the correct orientation of the key-tops. Each of the cues, as shown in table 5-1, is

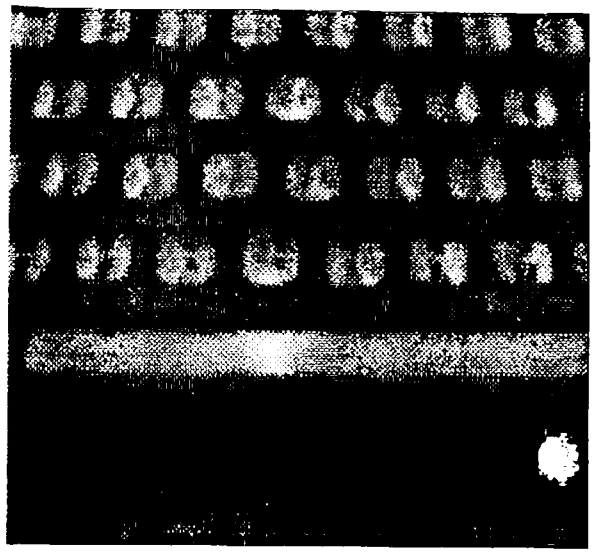


Figure 5-1: A computer terminal keyboard

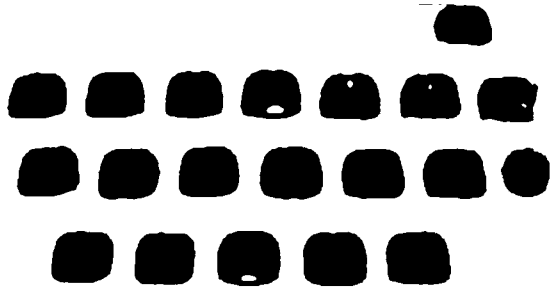


Figure 5-2: The texels of figure 5-1 mislead by the combination of noise and texel placement.

Yet the system, given the combination of the 5 cues, is able to generate the correct orientation of 18 of the 20 texels within the combination of the measurement errors (±2.75 degrees) and the accuracy of the tesselation (±1.21 degrees) (see figure 5-2). The p values of the remaining two texels are recovered within the combined approximations, while the q values have errors of 7.07 and

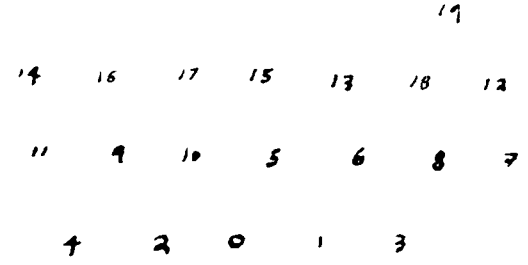


Figure 5-3: The numbering of the texels of figure 5-1

13.76 degrees. This error is due to the combination of noise and digitization errors as can be seen in figure 5-2.

5.1.2 Uniformly size coins

The next example consists of a surface textured with twelve uniformly sized coins (see figure 5-5). The system in this case ignores the incorrect orientation constraints from both the shape-from-uniform-texel-spacing and shape-from-parallel-virtual-lines cues and generates p and q values that are exact (see figure 5-4) for five of the twelve texel patches. One of the texels (number 10) has a correct q value and a p value with an error of 8.07 degrees, which is 2.5 degrees beyond the combined measurement error and tesselation approximation. The remaining six texels have an error of 9.5 degrees in their p value and a correct q value (see figure 5-7 and table 5-4). This error can be traced to a digitization error due to the reflectance of the coins. Specifically, the coins are metallic and reflect light differentially, causing portions of the coins to have lighter areas. Depending on the lighting conditions the threshold based blob finder may not consider the lightened areas as part of the blobs, thus decreasing the blobs shape and surface area. Furthermore, since the coins are raised above the surface they cast shadows. These shadow areas are, depending on the

Orientation in radians					
Texel #	shape-from-uniform-size	shape-from uniform-spacing	shape-from-parrallel-lines	shape-from-eccentricity	shape-from-eccentricty-change
0	p = 0.00 q = 0.67	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = 0.06 q = 0.95	unresolved
1	p = 0.00 q = 0.67	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = 0.11 q = 0.85	unresolved
2	p = -0.5 q = 0.45	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = -0.27 q = 0.55	unresolved
3	p = 0.00 q = 0.67	p = 0.00 q = -5.52	p = 0.00 q = -5.52	p = 0.17 q = 0.68	unresolved
4	p = 0.04 q = 0.63	p = 0.00 q = -5.52	p = 0.00 q = -5.52	p = 0.27 q = 0.37	unresolved
5	p = -0.5 q = 0.45	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = -0.17 q = 0.52	unresolved
6	p = -0.05 q = 0.54	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = -0.20 q = 20.17	unresolved
7	unresolved	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = -0.20 q = 0.48	unresolved
8	p = 0.00 q = 0.67	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = 0.15 q = 0.81	unresolved
9	p = 0.00 q = 0.67	p = 0.00 q = -5 52	p = 0.00 q = 5.52	0.15 q = -0.81	unresolved
10	p = 0.04 q = 0.63	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = 0.14 q = 0.52	unresolved
11	p = 0.00 q = 0.67	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = 0.24 q = 0.40	unresolved
12	p = -0.05 q = 0.54	p = 0.00 q = -5 52	p = 0.00 q = 5.52	p = -0.24 q = 0.59	unresolved
13	p = -0.04 q = 0.63	p = 0.37 q = 8.93	p = 0.00 q = 5.52	p = 0.27 q = 0.45	unresolved
14	p = 0.00 q = 0.67	p = 0.37 q = 8.93	p = 0.00 q = 5.52	p = 0.15 q = 0.81	unresolved
15	p = -0.02 q = 0.49	p = -0.37 q = 8.93	p = 0.00 q = 5.52	p = -0.20 q = 0.48	unresolved
16	p = -0.12 q = 0.51	p = 0.00 q = -5.52	p = 0.00 q = -5.52	p = 0.32 q = 0.56	unresolved
17	p = -0.05 q = 0.54	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = -0.20 q = 20.17	unresolved
18	p = -0.05 q = 0.54	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = .10 q = 0.55	unresolved
19	p = 0.00 q = 0.67	p = 0.00 q = -5.52	p = 0.00 q = 5.52	p = 0.14 q = 0.73	unresolved

Table 5-1: The 5 cues independently applied to the keyboard image

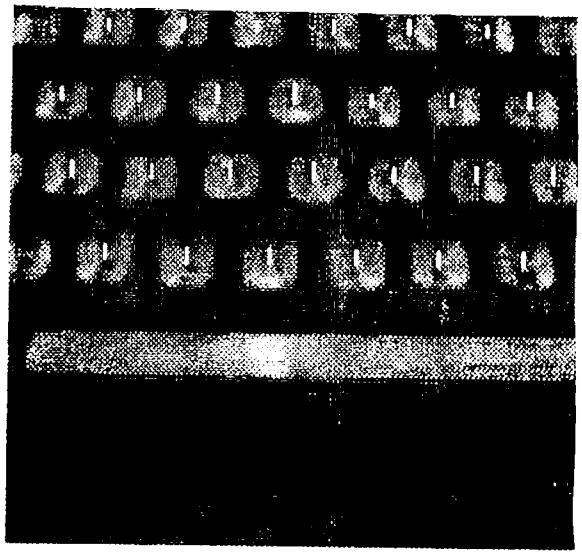


Figure 5-4: The surface normals for the image containing the computer keyboard

threshold levels, sometimes considered as part of the blobs.

5.1.3 IC Board

The second image is that of a simple PC board containing 14 sockets (see figure 5-8). The sockets are square in shape, uniformly sized, and uniformly spaced. The texel patch generator is able to approximately find the position and size of the sockets. Since the texels are positioned in two parallel lines (one group contains texels 1,3,5,7,9,11,13 and the other group contains texels 2,4,6,8,10,12,14) shape-from-uniform-spacing and shape-from-parallel-lines can only utilize vertical texel groupings. Therefore they are unable to recover any p information. Shape-from-uniform-size, shape-from-relative-eccentricity, shape-from-eccentricity are able to recover constraints in both the p and q directions.

After fusion the system recovers approximately the correct orientation for all of the 14 texels (see table 5-6).

Texel #	calculated value in radians	calculated value in degrees	measured value in radians	measured value in degrees	Error in Degrees
0	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
1	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
2	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
3	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
4	p = .024 q = 0.49	p = 1.4deg q = 26.02	p = 0 q = ??	p = 0 q = 30	1.4 3.98
5	p = .024 q = 0.49	p = 1.4deg q = 26.02	p = 0 q = ??	p = 0 q = 30	1.4 3.98
6	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
7	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
8	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
9	p = .05 q = 0.76	p = 2.9 q = 37.07	p = 0 q = ??	p = 0 q = 30	2.9 7.07
10	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
11	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
12	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
13	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
14	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
15	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
16	p = -.05 q = 0.29	p = -2.64 q = 16.24	p = 0 q = ??	p = 0 q = 30	2.64 13.76
17	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
18	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86
19	p = 0.0 q = 0.67	p = 0.0 q = 33.86	p = 0 q = ??	p = 0 q = 30	0 3.86

Table 5-2: The surface orientation values for the image containing the computer keyboard

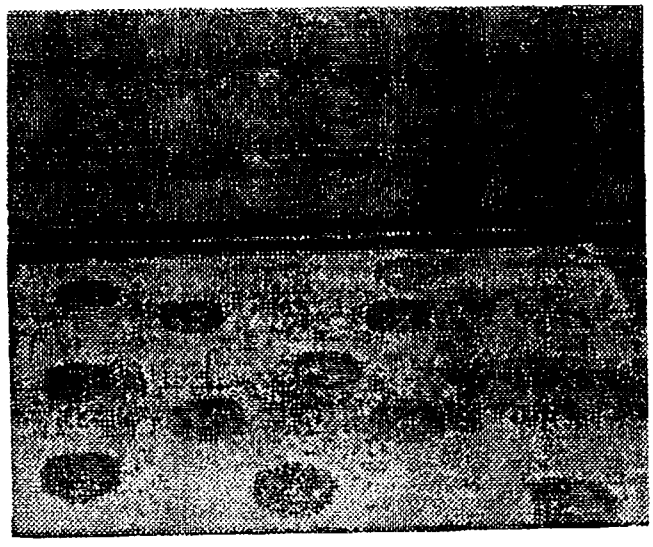


Figure 5-5: A surface covered with coins

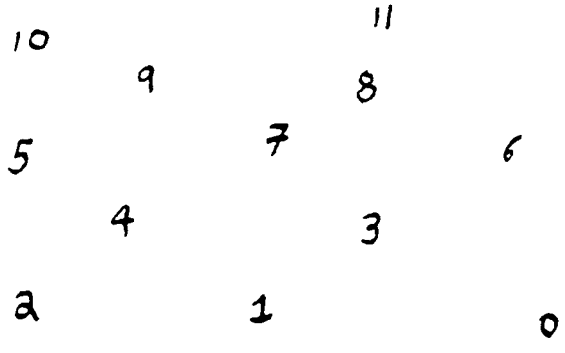


Figure 5-6: The numbering of the coins found

5.1.4 Surface covered with uniformly spaced geometric objects

The next image, figure 5-9, contains 8 uniformly spaced geometric objects. Shape-from-uniform-size and shape-from-relative-eccentricity are inapplicable to the image and, as shown in table 5-7, generate totally incorrect orientation information. Only shape-from-uniform-spacing and shape-from-parallel-lines are able to generate useful constraints for the whole image. Shape-from-eccentricity is able to generate useful information for the circle, two four-sided objects and the two squares. Yet after fusion the system is able to recover the correct orientation of all of the objects within the combined tesselation approximation and

Orientation in radians					
Texel #	shape-from-uniform-size	shape-from uniform-spacing	shape-from-parrallel-lines	shape-from-eccentricity	shape-from-eccentricty-change
0	p = 0.00 q = 3.05	unresolved	unresolved	p = 0.17 q = 3.96	unresolved
1	p = 0.00 q = 3.05	unresolved	unresolved	p = 0.14 q = 3.31	unresolved
2	p = 0.00 q = 3.05	unresolved	unresolved	p = 0.14 q = 3.31	unresolved
3	p = -0.14 q = 3.31	unresolved	unresolved	p = 0.14 q = 3.31	unresolved
4	p = 0.00 q = 3.05	unresolved	unresolved	p = 0.23 q = 2.62	unresolved
5	p = 0.00 q = 2.62	unresolved	unresolved	p = 0.14 q = 3.31	unresolved
6	p = -0.17 q = 3.96	unresolved	unresolved	p = 0.17 q = 3.05	unresolved
7	p = -0.17 q = 3.96	unresolved	unresolved	p = 0.17 q = 3.96	unresolved
8	p = -0.14 q = 3.31	unresolved	unresolved	unresolved unresolved	unresolved
9	p = -0.14 q = 3.31	unresolved	unresolved	p = 0.17 q = 3.96	unresolved
10	p = -0.26 q = 3.04	unresolved	unresolved	p = 0.00 q = 3.05	unresolved
11	p = 0.17 q = 3.96	unresolved	unresolved	p = 0.00 q = 2.62	unresolved

Table 5-3: The 5 cues independently applied to the surface covered with coins

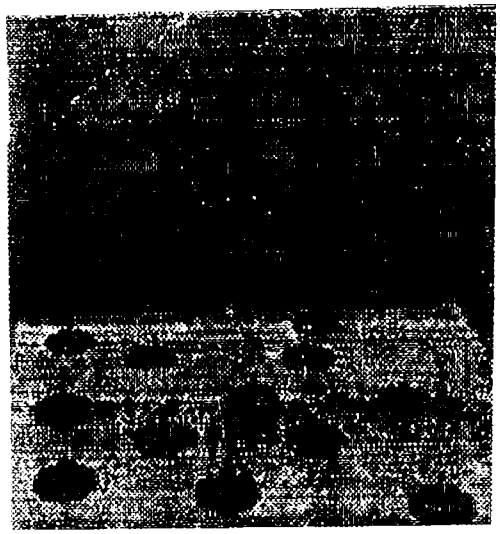


Figure 5-7: surface normals generated for the coins

measurement error.

Texel #	calculated value in radians	calculated value in degrees	measured value in radians	measured value in degrees	Error in Degrees
0	p = 0.2 q = 3.96	p = 9.52 q = 75.84	p = 0 q = 3.9	p = 0 q = 75.0	9.52 0.84
1	p = 0.0 q = 3.05	p = 0.0 q = 71.83	p = 0 q = 3.9	p = 0 q = 75.0	0 3.17
2	p = 0.0 q = 3.05	p = 0.0 q = 71.83	p = 0 q = 3.9	p = 0 q = 75.0	0 3.17
3	p = 0.0 q = 3.05	p = 0.0 q = 71.83	p = 0 q = 3.9	p = 0 q = 75.0	0 3.17
4	p = 0.0 q = 3.05	p = 0.0 q = 71.83	p = 0 q = 3.9	p = 0 q = 75.0	0 3.17
5	p = 0.0 q = 3.05	p = 0.0 q = 71.83	p = 0 q = 3.9	p = 0 q = 75.0	0 3.17
6	p = -0.2 q = 3.96	p = -9.52 q = 75.84	p = 0 q = 3.9	p = 0 q = 75.0	9.52 0.84
7	p = 0.2 q = 3.96	p = 9.52 q = 75.84	p = 0 q = 3.9	p = 0 q = 75.0	9.52 0.84
8	p = -0.2 q = 3.96	p = -9.52 q = 75.84	p = 0 q = 3.9	p = 0 q = 75.0	9.52 0.84
9	p = 0.2 q = 3.96	p = 9.52 q = 75.84	p = 0 q = 3.9	p = 0 q = 75.0	9.52 0.84
10	p = -0.14 q = 3.31	p = -8.07 q = 73.19	p = 0 q = 3.9	p = 0 q = 75.0	8.07 1.81
11	p = -0.2 q = 3.96	p = -9.52 q = 75.84	p = 0 q = 3.9	p = 0 q = 75.0	9.52 0.84

Table 5-4: Orientation values for the image containing coins

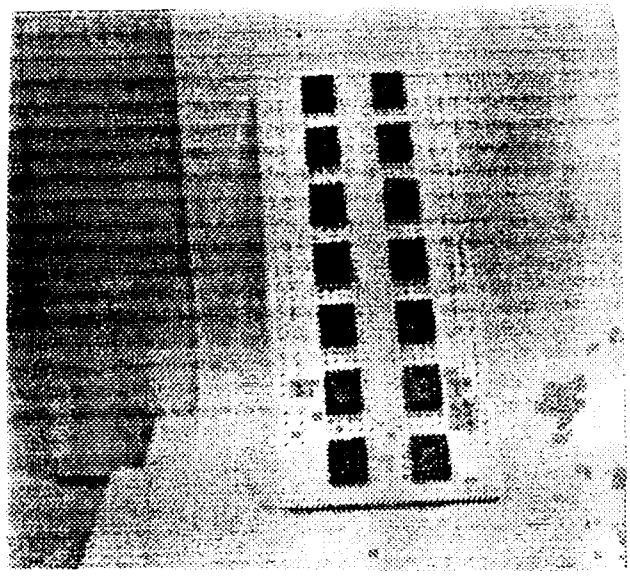


Figure 5-8: A PC board with 14 sockets

Orientation in radians					
Texel #	shape-from-uniform-size	shape-from uniform-spacing	shape-from-parrallel-lines	shape-from-eccentricity	shape-from-eccentricty-change
0	p = -0.14 q = 0.73	unresolved q value	unresolved q value	p = -0.16 q = 0.00	unresolved
1	p = -0.08 q = 0.73	unresolved q value	unresolved q value	p = -0.16 q = -0.04	unresolved
2	p = -0.8 q = 0.65	unresolved q value	unresolved q value	p = -0.38 q = -0.08	unresolved
3	p = -0.08 q = 0.65	unresolved q value	unresolved q value	p = 0.18 q = -0.23	unresolved
4	p = -0.09 q = 0.72	unresolved q value	unresolved q value	p = -0.16 q = -0.04	unresolved
5	p = -0.08 q = 0.65	unresolved q value	unresolved q value	p = -0.18 q = 0.0	unresolved
6	p = -0.08 q = 0.65	unresolved q value	unresolved q value	p = -0.16 q = -0.04	unresolved
7	p = -0.04 q = 0.63	unresolved q value	unresolved q value	p = -0.04 q = 0.07	unresolved
8	p = -0.08 q = -0.65	unresolved q value	unresolved q value	p = -0.16 q = 0.04	unresolved
9	p = -0.08 q = -0.65	unresolved q value	unresolved q value	p = -0.30 q = -0.04	unresolved
10	p = -0.08 q = 0.65	unresolved q value	unresolved q value	p = -0.25 q = 0.04	unresolved
11	p = -0.08 q = -0.65	unresolved q value	unresolved q value	p = -0.16 q = -0.04	unresolved
12	p = -0.22 q = 0.70	unresolved q value	unresolved q value	p = -0.16 q = 0.04	unresolved
13	p = -0.14 q = 0.73	unresolved q value	unresolved q value	p = -0.11 q = -0.19	unresolved

Table 5-5: The 5 cues independently applied to the IC Board

5.2 Multiple Surfaces

The next group of images show the system's ability to recover the correct orientation even when the image contains multiple surfaces. As stated earlier, the system has no a priori information as to the number of surfaces.

5.2.1 2 Surfaces Covered with Nickels

The first example of an image containing multiple surfaces is shown in figure 5-10. The image contains 2 surfaces which are covered with a total of seventeen nickels, eight on one surface and nine on the other. This is a very

Texel #	calculated value in radians	calculated value in degrees	measured value in radians	measured value in degrees	Error in Degrees
0	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
1	p = 0.0 q = 5.52	p = 0.0 q = 77.14	p = 0 q = 4.9	p = 0 q = 78.5	0 1.36
2	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
3	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
4	p = .024 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	1.4 1.23
5	p = .024 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	1.4 1.23
6	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
7	p = 0.0 q = 5.52	p = 0.0 q = 77.14	p = 0 q = 4.9	p = 0 q = 78.5	0 1.36
8	p = 0.47 q = 0.61	p = 25.28 q = 31.42	p = .5 q = .8	p = 26.6 q = 38.6	1.32 7.18
9	p = 0.59 q = 0.77	p = 30.36 q = 37.71	p = .5 q = .8	p = 26.6 q = 38.6	3.76 0.89
10	p = .049 q = .81	p = 26.15 q = 38.85	p = .5 q = .8	p = 26.6 q = 38.6	0.45 0.25
11	p = .0.50 q = 0.68	p = 26.60 q = 34.15	p = .5 q = .8	p = 26.6 q = 38.6	0 4.45
12	p = 0.59 q = 0.77	p = 30.36 q = 37.71	p = .5 q = .8	p = 26.6 q = 38.6	3.76 0.89
13	p = 5..5 q = 5.67	p = .5. 5 q = 33.86	p = .5 q = .8	p = 26.6 q = 38.6	0 0
14	p = 0.59 q = 0.77	p = 30.36 q = 37.71	p = .5 q = .8	p = 26.6 q = 38.6	3.76 0.89
15	p = 0.55 q = 0.83	p = 28.94 q = 39.67	p = .5 q = .8	p = 26.6 q = 38.6	2.34 1.07
16	p = 0.47 q = .61	p = 25.28 q = 31.42	p = .5 q = .8	p = 26.6 q = 38.6	1.32 7.18

Table 5-6: Orientation values for the image containing IC board

difficult image containing a number of real world problems that would be unsolvable by a less robust approach. These problems include:

1. The nickels. in the image have thickness and are raised slightly above the surface creating slight shadows. These shadows are considered by the threshold based blob finding algorithm as part of the texels, thus generating incorrect sizes for the blobs. This in turn can cause the shape-from-uniform-texel-size algorithm to generate inaccurate results.

2. The nickels are shiny and reflect the lighting. This can cause parts

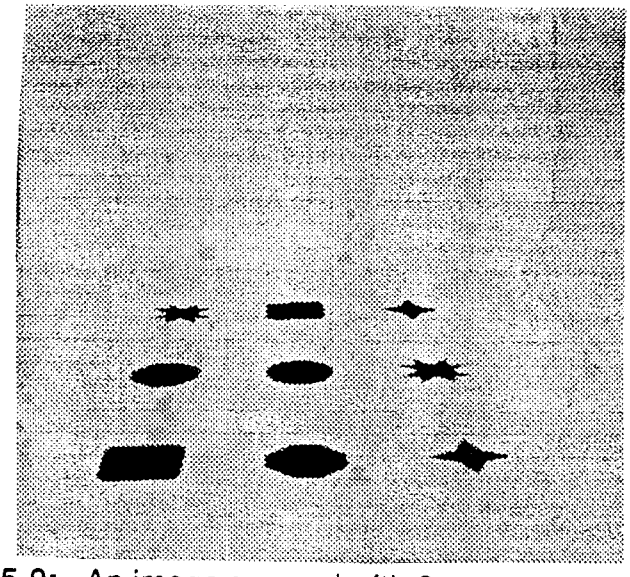


Figure 5-9: An image covered with 8 geometric objects

Orientation in radians					
Texel #	shape-from-uniform-size	shape-from-uniform-spacing	shape-from-parrallel-lines	shape-from-eccentricity	shape-from-eccentricty-change
0	p = 1.03 q = 0.8.88	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = 0.06 q = 0.95	unresolved
1	p = 1.03 q = 0.8.88	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = 0.06 q = 0.95	unresolved
2	p = 1.29 q = 2.12	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = -0.27 q = 0.55	unresolved
3	p = 1.03 q = 0.8.88	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = 0.06 q = 0.95	unresolved
4	p = 0.04 q = 0.63	p = 0.00 q = 4.38	p = 0.00 q = -5.52	p = 0.27 q = 0.37	unresolved
5	p = -0.5 q = 0.45	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = -0.17 q = 0.52	unresolved
6	p = 0.37 q = 8.93	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = -0.20 q = 20.17	unresolved
7	unresolved	p = 0.00 q = 4.38	p = 0.00 q = -5.52	p = 0.00 q = 0.48	unresolved
8	p = 1.21 q = 6.26	p = 0.00 q = 4.38	p = 0.00 q = 5.52	p = 0.15 q = 0.81	unresolved

Table 5-7: The 5 cues independently applied to the Geometric Objects

Texel #	calculated value in radians	calculated value in degrees	measured value in radians	measured value in degrees	Error in Degrees
0	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
1	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
2	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
3	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
4	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
5	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
6	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64
7	p = 0.0 q = 3.14	p = 0.0 q = 72.36	p = 0 q = 3.5	p = 0 q = 76	0 3.64

Table 5-8: The 5 cues independently applied to the geometric objects

of a texel to be considered as background, causing the method to ignore pixels that should be part of texels.

3. The texels are randomly placed, yet some accidental alignments of texels do occur. In fact, texels 3, 9, and 12 are sufficiently co-linear to cause shape-from-uniform-texel-spacing and shape-from-parallel-virtual-lines to generate orientation constraints of relatively high weighting.

4. To add to these imaging difficulties are the more general problems that arise because there is more than one surface, and the system has no knowledge about the placement of surfaces. Furthermore, some of the texels of the different surfaces are closer to each other than they are to texels of their own surface (e.g. texel 7 and texel 8 in figure 5-10).

If the cues are utilized independently then the severity of these errors becomes apparent. This is shown in table 5-9.

After computing the valid constraints the system is able to determine that shape-from-uniform-texel-spacing and shape-from-virtual-parallel-lines generated none of the valid constraints. For texel 3, the system believes that it is uniformly spaced, yet no other texels are considered as uniformly spaced so the texture analysis phase ignores this information. The valid constraints are due to shape-

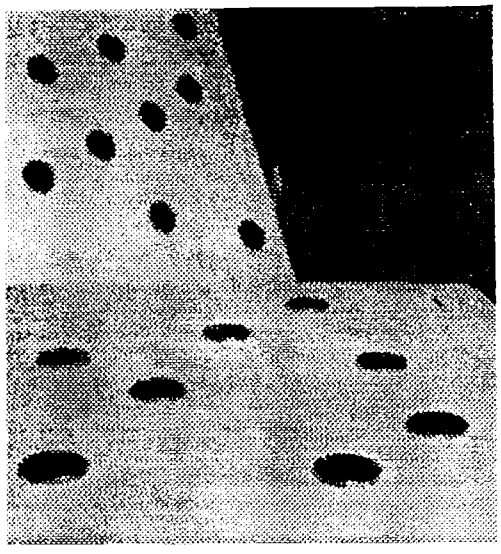


Figure 5-10: An image with two surfaces covered with nickels

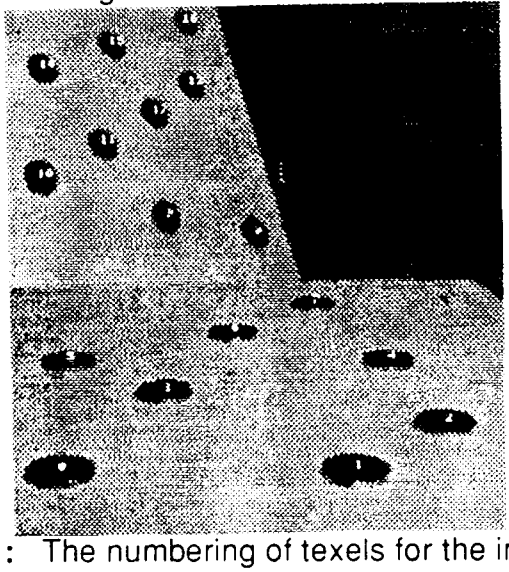


Figure 5-11: The numbering of texels for the image with two surfaces covered with nickels

from-uniform-texel-size, shape-from-relative-eccentricity, and shape-from-eccentricity.

Considering these potential problems the system responds well[15]. The surface orientations were measured within ± 2.75 degrees and the computed orientations was generated by the system to within ± 1.21 degrees. The system

[15]The orientation results shown here differ from those of [Moerdler and Kender 87b] because the system described here contains an additional shape-from-texture cue and has an automated histogram thresholding algorithm that generates the texels. The combination of these two factors cause the system to generate different, and in most cases, superior results.

Orientation in radians					
Texel #	shape-from-uniform-size	shape-from uniform-spacing	shape-from-parrallel-lines	shape-from-eccentricity	shape-from-eccentricty-change
0	p = 0.00 q = 5.52	unresolved	unresolved	p = 4.14 q = -5.01	unresolved
1	p = 0.00 q = 4.38	unresolved	unresolved	p = 0.00 q = 4.38	unresolved
2	p = 0.00 q = 5.52	unresolved	unresolved	p = 0.98 q = -1.28	unresolved
3	p = 0.46 q = 0.73	unresolved	unresolved	p = -0.50 q = -3.97	unresolved
4	p = 0.00 q = 4.38	unresolved	unresolved	p = -1.66 q = -8.80	unresolved
5	p = 0.00 q = 5.52	unresolved	unresolved	p = -1.09 q = -3.85	unresolved
6	p = 0.00 q = 4.38	unresolved	unresolved	p = -7.96 q =-9.38	unresolved
7	p = 0.00 q = 4.38	unresolved	unresolved	p = 6.57 q = 5.02	unresolved
8	p = 0.59 q = 0.77	unresolved	unresolved	p = 0.26 q = -6.33	unresolved
9	p = 0.59 q = 0.77	unresolved	unresolved	p = 0.37 q = -8.93	unresolved
10	p = 0.50 q = 0.68	unresolved	unresolved	p = 3.29 q = -4.26	unresolved
11	p = 0.50 q = 0.68	unresolved	unresolved	p = 3.29 q = -4.26	unresolved
12	p = 0.50 q = 0.68	unresolved	unresolved	p = 3.18 q = -4.53	unresolved
13	p = 0.30 q = 0.67	unresolved	unresolved	p = 3.29 q = -4.26	unresolved
14	p = 0.50 q = 0.68	unresolved	unresolved	p = 2.31 q = -3.12	unresolved
15	p = 0.30 q = 0.68	unresolved	unresolved	p = -4.26 q = -8.53	unresolved
16	p = 0.39 q = 0.64	unresolved	unresolved	p = 0.83 q = 0.26	unresolved

Table 5-9: The 5 cues independently applied to the 2 nickels image computed an orientation within the measurement error for 9 of the seventeen texels. Of the remaining 8 texels 6 texels (10,11,12,13,14,15) were within ±4.5 degrees which is within the combined error due to the measurement error and the computation error (see table 5-10). For texels 3 and 16 the major component of the measurement error is due to the difficulty the texel patch generator had in recovering the texels. Texel 3 was also part of an accidental co-linear grouping; it

Figure 5-12: Surface Normals for the image containing two surfaces covered with nickels

caused the shape-from-uniform-spacing and shape-from-parallel-virtual-line cues to generate incorrect constraints.

5.3 Summary

This chapter has shown the experimental results of the complete multiple shape-from-textures system. A total of 6 images, five in this chapter and one in the previous chapters, have been given that show how the fusion of 5 different cues can result in the recovery of surface orientation information for a wide range of textured surfaces. The system works even when three of the five cues generate incorrect constraints (e.g. figure 5-9). Additional research results point to the system being able to correctly recover surface orientation information even when there exist more than one texture on the surface and when the textures are interspersed. Future research should be directed at defining how effect this approach is for multiply textured surfaces and multiple overlapping surfaces.

Texel #	calculated value in radians	calculated value in degrees	measured value in radians	measured value in degrees	Error in Degrees
0	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
1	p = 0.0 q = 5.52	p = 0.0 q = 77.14	p = 0 q = 4.9	p = 0 q = 78.5	0 1.36
2	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
3	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
4	p = .024 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	1.4 1.23
5	p = .024 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	1.4 1.23
6	p = 0.0 q = 5.52	p = 0.0 q = 79.73	p = 0 q = 4.9	p = 0 q = 78.5	0 1.23
7	p = 0.0 q = 5.52	p = 0.0 q = 77.14	p = 0 q = 4.9	p = 0 q = 78.5	0 1.36
8	p = 0.47 q = 0.61	p = 25.28 q = 31.42	p = .5 q = .8	p = 26.6 q = 38.6	1.32 7.18
9	p = 0.59 q = 0.77	p = 30.36 q = 37.71	p = .5 q = .8	p = 26.6 q = 38.6	3.76 0.89
10	p = .049 q = .81	p = 26.15 q = 38.85	p = .5 q = .8	p = 26.6 q = 38.6	0.45 0.25
11	p = .0.50 q = 0.68	p = 26.60 q = 34.15	p = .5 q = .8	p = 26.6 q = 38.6	0 4.45
12	p = 0.59 q = 0.77	p = 30.36 q = 37.71	p = .5 q = .8	p = 26.6 q = 38.6	3.76 0.89
13	p = 5.5 q = 5.67	p = 5.5 q = 33.86	p = .5 q = .8	p = 26.6 q = 38.6	0 0
14	p = 0.59 q = 0.77	p = 30.36 q = 37.71	p = .5 q = .8	p = 26.6 q = 38.6	3.76 0.89
15	p = 0.55 q = 0.83	p = 28.94 q = 39.67	p = .5 q = .8	p = 26.6 q = 38.6	2.34 1.07
16	p = 0.47 q = .61	p = 25.28 q = 31.42	p = .5 q = .8	p = 26.6 q = 38.6	1.32 7.18

Table 5-10: Orientation values for the image containing two surfaces covered with nickels

6. Extensions to the Basic Paradigm

Texture classification, surface segmentation, and the recovery of shape from texture are known hard problems for which many theories have been developed. In fact, no theory that explains any one of these three processes would be complete without dealing with the interaction between the three, because texture is a product of numerous physical processes [Laws 80].

This chapter describes an approach that effectively integrates texture classification and segmentation with the recovery of the surface shape parameters. In the approach, the multiple shape-from-texture paradigm that has been developed in this work is used to derive the shape of the surfaces in the image, while separately constraining texture classification and surface segmentation.

The multiple shape-from-texture paradigm can aid in texture classification because of the way in which shape-from-texture cues function. As described previously in this work, texture can be used to derive the shape of surfaces if apriori assumptions are made about the surface texture. Individually, shape-from-texture cues are limited by their underlying assumptions, yet if integrated into a large system comprised of multiple cues the surface orientation can be recovered. The specific cues that are used to derive the surface orientation can be found, and their underlying assumptions used to restrict the class of possible textures. If a large enough group of shape-from-texture methods is used, then the texture classification problem becomes greatly constrained (as per chapter 2). This will be discussed in greater depth in section 6.2.

The surface segmentation problem can be simplified using information or texture classifications generated during the operation of a multiple shape-from-texture system. Once the surface parameters have been recovered the constraints can be re-analyzed and the texels that were used to generate the surface parameters can be combined into groupings. As will be shown below, these groups are a first approximation to surface segmentation. Additional

segmentation information can be supplied by the texture classification component, if the various surfaces in the image differ in their textures.

6.1 Recovering Valid Orientation Constraints

In the previous two chapters the steps necessary for the multiple shape-from-texture system to generate surface orientation information per texel patch has been derived. In order to recover surface segmentation and texture classification information, the system must re-analyze the orientation constraints to recover the "valid constraints". Specifically, the term "valid constraints" is taken to denote those constraints that were used to generate a solution it is not meant to imply any world knowledge that the constraints are correct.

The valid orientation constraints are determined individually for each surface patch by comparing each orientation constraint to the texel patch's orientation. The validity of each constraint can be simply determined by checking to see if the constraint falls on the chosen trixel. It should be noted that the constraint is considered valid up to the approximateness with which the orientation constraint was originally calculated.

6.2 Texture Classification

Texture classification is normally performed by computing which of a group of features describe a given texture. No single set of features can differentiate all possible textures [Laws 80]. Instead, researchers have proposed different types of features for specific classes of textures (e.g. autocorrelation functions, etc.). Many of the features that have been proposed are ad hoc.

At first the relationship between these texture classification methods and a system based on the multiple shape-from-texture paradigm would seem slim. Yet a careful examination will show that they are closely related. Previous texture classification methods measured texture properties between images or areas of the image in much the same way as shape-from-texture cues do. Furthermore,

these methods are most applicable when each image area is large in relationship to the texture (large is defined differently for each method) and the surface is locally planar, criteria similar to those of the multiple shape-from-texture system.

At least two different types of classification methods exists: binary-image-based and grey-level image-based. The shape-from-texture cues in this work are binary image based and therefore are most similar to the binary image based classification methods. If the relationship between texture classification and shape-from-texture that is proposed here is valid, then grey-level based shape-from-texture cues would also be worth considering.

The similarity between the specific texture properties that are used by the binary image based classification methods and those of shape-from-texture cues can be understood by considering a selection of these classification methods: structural element approaches, autocorrelation functions, and optical and digital transforms.

The structural element approach [Serra 74] detects "regular elements" and measures the spatial regularity. These "regular elements" which are sufficiently large to be individually measured can be equated with texels. The regularity measure is thus similar to shape-from-uniform-size.

Autocorrelation functions, optical transforms, and digital transforms all measure spatial frequency either directly or indirectly [Haralick 78]. These spatial frequency measures are similar to shape-from-uniform-spacing if the elements are large, and shape-from-density or shape-from-autocorrelation if the spatial frequency is determined as edges per unit area.

Most of these methods have two major disadvantages which limit their applicability. First, they are normally global in nature, relying upon a global classification of image texture. This is a major problem since more real world images are comprised of multiple textures and multiple surfaces. When the image contains multiple textures and/or multiple surfaces the global

transformation combines all of the information into a single result making the classification and segmentation difficult or impossible.

Second, they assume that the textured surface is purely planar. Even if the structure is successfully recovered the resulting texture description is an average texture pattern rather than the true underlying texture.

Recently researchers [Hamey 88] have attempted to decouple the texel recovery stage from the texture classification stage. Hamey defines a method for recovering the texture of repetitively textured surfaces. Thus local texture information can be recovered. By removing the global consistency assumption, and making no a priori assumption about the frequency of the texels, a paradox arises: texel information is required to recover the frequency, and frequency information is required to recover the texels.

Given the similarities between texture classification and multiple shape-from-texture a simple addition to the multiple shape-from-texture paradigm allows it to not only recover surface orientation but also some primative texture classification information.

The underlying assumptions of existing shape-from-texture cues limit the class of textures to which the cue is applicable. If it can be established which cue is appliable to a specific texture, it is in effect equivalent to establishing the texture classification. Since this classification information is unavailable prior to deriving the surface shape, the multiple shape-from-texture paradigm applies all of the cues to the image. After the knowledge fusion phase the system is able to determine which are the valid constraints and therefore which assumption are valid for each texel.

The underlying assumptions of shape-from-texture cues attempt to model the intrinsic properties of texture (e.g. uniform space, uniform size etc. [Gibson 50]). The texture classification algorithm is therefore based on a model of the texture. For example, shape-from-uniform-texel-spacing models the same

intrinsic texture property as texture classification based on spatial frequency.

The inter-relationship between surface orientation recovery, surface segmentation, and texture classification is important and should be used to generate a consistant solution to all three. For example, the surface consistency information that is generated by shape-from-texture can be used in the surface segmentation and texture classification. Likewise, the texture classification information can be used to prune the shape-from-textures cues that are used thus decreasing the amount of noise and helping to generate a more exact solution.

If a large number of cues utilize different intrinsic properties of the texture, then the texture model becomes complex enough to be able to describe both natural and synthetic texture.

The point is that if there is a consistent surface (even one covered with more than one overlapping texture) then at least one of the cues will be applicable.The texture can be classified by which cues are applicable and which cues are not.

Laws first order texture energy transform [Laws 80] utilizes a group of convolution masks that measure specific features such as 'edge-like' and 'v-shape-like'. Effectively, each of these features can be equated to different texture patch generator definitions. If a system based on the multiple shape-from-texture paradigm is modified to fuse the results of multiple texel patch generators, then the system will also be able to measure which texel patch generators are effective for the image, and thus which features the texture contains.

If surface segmentation is performed at the same time as texture classification then not only can the image be partitioned into regions but a texture model can be generated for each region in the image.

6.3 Deriving of Surface Segmentation Information

The final phase of the system generates surfaces from the individual augmented texels. This is done by re-analyzing the orientation constraints generated by the shape-from methods, determining which augmented texels are part of the same surface. This is effectively a first approximation of surface separation and segmentation.

The re-analysis consists of iterating through each augmented texel, considering all its orientation constraints, and determining which constraints aided in defining the "most likely" orientation for the texel patch. If an orientation constraint correctly determined the orientation of all the texels that were used in generating the constraint, then these augmented texels are considered as part of the same surface. This information can be fed to the surface segmentation phase to aid the segmentation process. The resulting sub-images (each containing a single surface) can then be fed to the multiple shape-from-texture system to generate a more exact surface orientation for each of the surface patches.

Some of the previous texture based segmentation algorithms [Kjell and Dyer 86; Raafat and Wong 86], were based on similar principles to our method, they partitioned images into regions based on differences in the imaged texture. The difference is defined as a measured change in some feature or features of the texture (also called texture measures). Depending on the specific group of features used, an image may be segmented into different regions.

Since these methods use only information based on texture classification to partition the image, the segmented regions do not necessarily correspond to surfaces. If the features are either too sensitive or do not really model the world then surfaces will be partitioned into multiple regions. At the other extreme, if the features do not correspond to all of the attributes of the texture then regions of the image may contain more than one surface. An additional handicap of these texture segmentation algorithms is that they are unable to correctly partition images containing overlapping transparent textured surfaces.

Perceived texture is the product of numerous physical processes. If segmentation is to be more exact, then it should consist of more than measured changes in attributes of the texture. The multiple shape-from-texture paradigm can easily be extended to aid in surface segmentation.

6.4 Examples

To better understand how the system performs texture classification and surface segmentation let us consider a couple of examples.

6.4.1 Computer Terminal Keyboard

Subsection 5.1.1 discussed how the system solved the orientation of a real camera-acquired image containing a computer terminal keyboard. This figure, repeated below, is a good example of the system's ability to supply texture classification information.

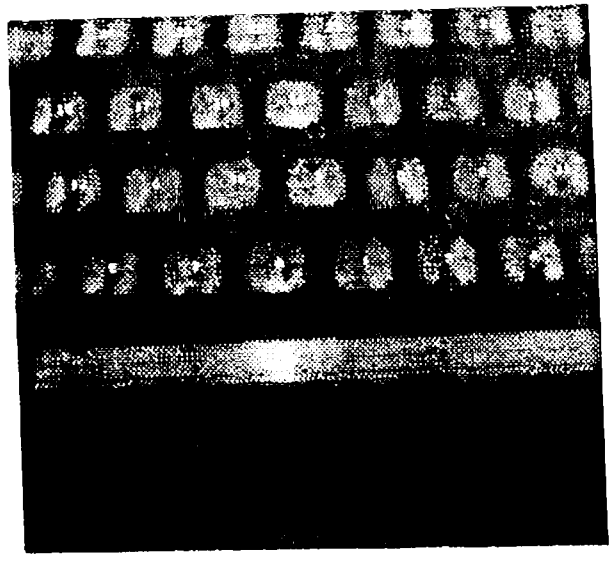


Figure 5-1: A computer terminal keyboard

Once the orientation information is recovered, the texture classification step determines that all of the texels in the image are uniformly sized, uniformly spaced horizontally, have axis of symmetry with equal moments, and are located

on parallel horizontal lines. This information, which is easily recovered by the integrated approach, strongly constraints the texture classification problem. Furthermore, if more shape-from-texture cues were to be added to the system, texture classification would be able to generate additional texture classification information.

The surface segmentation step groups the texels of figure 5-1 into four separate horizontal, groupings based on information generated by shape-from-uniform-texel-spacing and shape-from-virtual-parallel-lines. Shape-from-uniform-texel-size supplies additional information that allows the horizontally grouped texels to be combined into a single surface. The system is able to recover a single surface where many purely feature based segmentation methods, such as some spatial features methods, would not.

6.4.2 2 Surfaces Covered with Nickels

Figure 5-10 contains two surfaces covered with nickels. The orientation information recovered for this image was discussed in subsection 5.2.1.

Figure 5-10: An image with two surfaces covered with nickels

The system is able to group almost all of the texels correctly, with texels 0,1,3,2,4,5,6,7 as part of one surface and texels 8,9,10,11,12,13,14,15,16,17 as

part of the second surface. Texel 13, due to errors described above, was considered by the system as part of neither surface. If additional shape-from-texture cues are added to the system texel 13 might be linked correctly into the lower surface.

The system's ability to segment the image results from the grouping information supplied by the valid constraints and without any a priori segmentation information[16]. Many previous texture based segmentation algorithms would probably try to group the surfaces into a single surface because the change in statistical texture features do not change sufficiently across the surface boundaries.

Figure 5-12: Surface Normals for the image containing two surfaces covered with nickels

After computing the valid constraints the system is able to determine that shape-from-uniform-texel-spacing and shape-from-virtual-parallel-lines generated none of the valid constraints. For texel 13 the system is unable to recover classification information. The only valid constraints are due to shape-from-uniform-texel-size, shape-from-eccentricity-change, and shape-from-eccentricity. This information allows the texture classification component to determine

[16]The only segmentation information is generated by the heuristic blob finder and is used to seperate the image into blobs

separately for each of the two surfaces that the textures are comprised of uniformly sized textural elements whose moments are equal (due to shape-from-eccentricity), and are not uniformly spaced or located on virtual parallel lines.

6.5 Summary

In this chapter the multiple shape-from-texture paradigm has been extended to aid in both surface segmentation and texture classification. The extensions are based on the fact that during constraint generation phase the system generates large numbers of constraints that are based on both different texture models (due to the different texture cues used) and different groupings of texels. By determining which models were applicable, the system recovers texture classification information. Surface segmentation is approximated by grouping texels together that have identical valid constraints.

7. Conclusions

In this work a new approach to the problem of deriving surface parameters has been proposed, and experimentally validated. A fusion paradigm has been described (see figure 1-3) which defines how a group of homeogeneous cues can be fused together to create a more powerful whole.

The paradigm has been applied to the problem of integrating multiple shape-from-texture cues. A four phase system has been created consisting of: the generation of *texel patches* (here, the proper level of abstraction), the calculation of orientation constraints by the independent cues, the consolidation of constraints into a "most likely" orientation per patch, and generation of surface segmentation and texture classification information.

During the first phase, the different shape-from-texture components generate texel patches and *augmented texels*. Each augmented texel consists of the 2-D description of a texel patch and a list of weighted constraints on its orientation. The orientation constraints for each patch are potentially inconsistent. This occurs due to many reasons: the shape-from methods may be applied to noisy real images, they are locally based; they derive constraints without a priori knowledge of the type of texture or the number of surfaces.

Each constraint is given an intra-cue weight which attempts to model the probability that the constraint is "more probably" correct. This weight is based on major underlying assumptions of the cue, such as the fact that the texels used to generate the constraint are on a planar surface patch.

The constraint weightings are normalized between cues. This normalization is important since different cues generate different numbers of constraints.

In the second phase, all the orientation constraints for each augmented texel are consolidated into a single "most likely" orientation by a Hough-like accumulation over a tesselated Gaussian sphere. During this phase the system

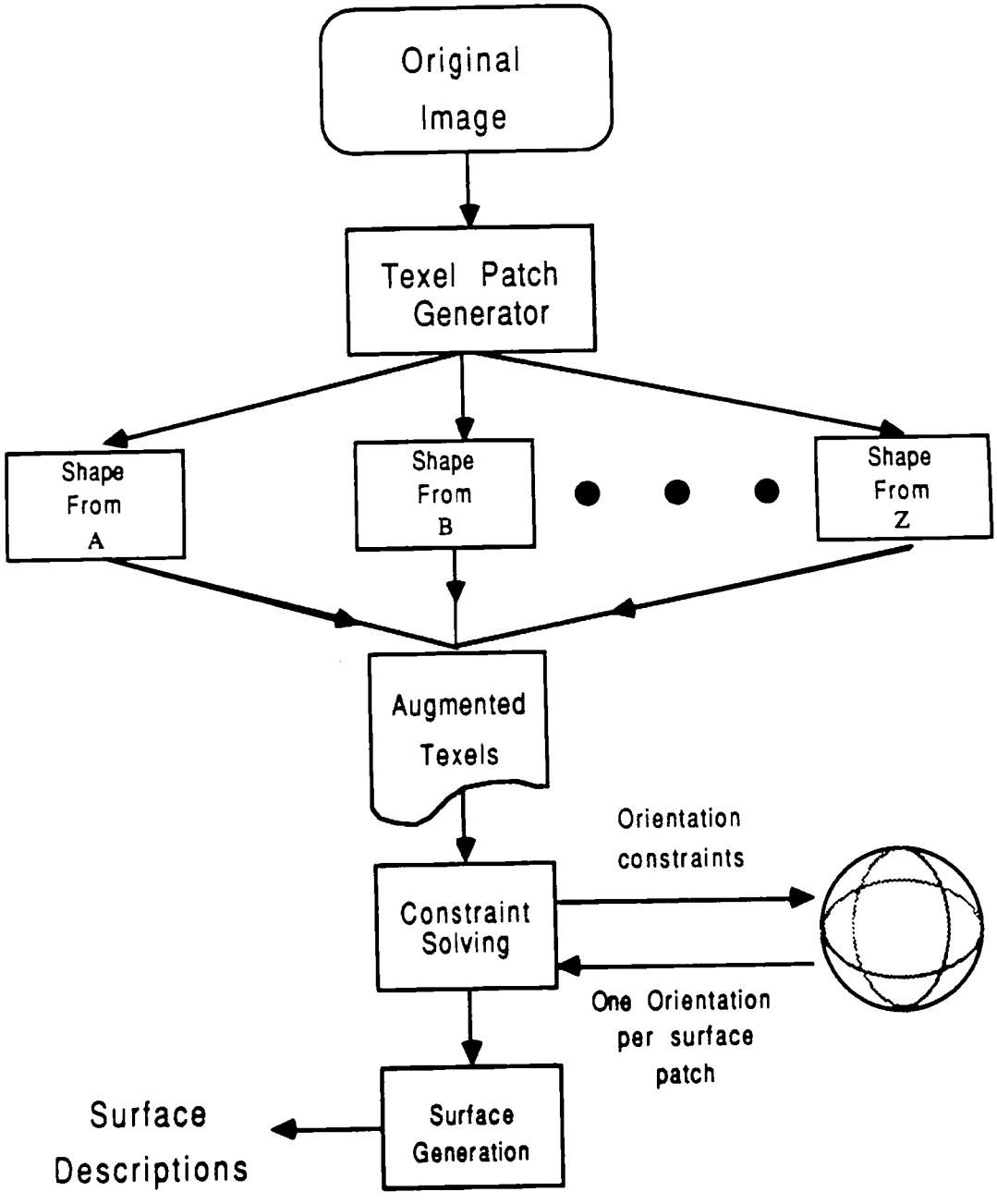


Figure 1-3: Integrating multiple shape-from methods

will also logically merge together all the augmented texels that cover the same locality ("surface patch") of the image. This is possible since many of the shape-from components define "texel" similarly, thus the constraints generated should be merged, and a single orientation generated for the surface patch.

To assure that the best orientation is choosen for each texel patch the

system iterates through all of the texel patches and chooses the patch which has the highest orientation weighting. Once the patch is choosen its orientation is fixed, and the system utilizes this texel patch to solve the other patches. This system iterates through all of the texel's constraints, and removes all constriants that were not used to generate the patch's "most likely" orientation. These non-valid constraints are used to prune the other texel patches' constraints. Since most constraints are generated utilizing more than one texel patch, the system is able to remove constraints from the unsolved texel patches. In short, this is a type of relaxation applied to the "intrinsic image" of surface orientation.

Finally, the system re-examines for each orientation constraint those augmented texels that are part of the same constraint family and groups them together by surface orientation. In effect, this segments the image into regions of similar orientation. The system also determines which cues were used to solve for the surface orientations and is able to perform a simple texture classification from this information.

7.1 Contribution of This Research

This work is new in ten ways:

1. The system has been applied to a wide range of real imagery. The images are real world images in which few a priori assumptions are made about the number of surfaces in the image, the type of texture (within a fairly wide class), the placement of textural elements, or the curvature of the surface(s).
2. The paradigm allows for the integration of cues, some of which are statistical while others are structural. This "blurring" of the statistical/structural boundary increases the overall applicability of the paradigm.
3. Existing shape-from-texture cues have been analyzed and found to be composed of two major constituent components: their definition of a texel, and their specific orientation recovery algorithm. This separation simplifies the process of defining a range of new shape-from-texture cues. New cues can be created by taking any of the texel definitions and combining it with any of the recovery algorithms.

4. A robust shape-from-texture system has been built and demonstrated, that is based upon the integration of multiple methodologies that cooperate and/or conflict.

5. Because the shape-from-texture system solves the orientation of each texel patch independently it is able to restrict the effects of noise to the specific texels that are either generated by noise or directly affected by the noise.

6. A new data structure, the augmented texel, was created in order to simplify the fusion of multiple orientation constraints. Furthermore, this data structure allows for a simple description of many of the salient features of surface patches.

7. A two level constraint weighting scheme has been created that allows the integration of orientation constraints derived by different methodologies, even under conditions in which one or more of the methods generate noisy or incorrect constraints.

8. Since the method fuses constraints per texel patch the method is shown to aid in the segmentation of surfaces.

9. The method aids in the separation of transparent surfaces.

10. Once all of the texel patch orientations are recovered the system can determine which cues were applicable to the surfaces, and thus aid in the classification of the type of texture on the surface(s): fusion leads to segmentation and recognition.

References

[Allen 85] Peter K. Allen.
Object Recognition Using Vision and Touch.
PhD thesis, University of Pennsylvania, Department of Computer Science, 1985.

[Aloimonos 86] John Aloimonos.
Detection of Surface Orientation and Motion from Texture: 1. The Case of Planes.
In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1986.

[Aloimonos and Chou 85]
John Aloimonos and Paul B. Chou.
Detection of Surface Orientation and Motion from Texture: 1. The Case of Planes.
Technical Report, University of Rochester, January 1985.

[Aloimonos and Swain 85]
John Aloimonos and Michael J. Swain.
Shape from Texture.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* IJCAI, IJCAI, 1985.

[Bajcsy and Lieberman 76]
R. Bajcsy and L. Lieberman.
Texture Gradient as a Depth Cue.
Computer Graphics and Image Processing 5:52-67, 1976.

[Ballard and Brown 82]
Dana Ballard and Christopher Brown.
Computer Vision.
Prentice-Hall Inc., 1982.

[Bandopadhyay 84]
A. Bandopadhyay.
Interesting Points, Disparities, and Correspondence.
In *Proceedings of the Image Understanding Workshop*, pages 184-187. DARPA, Scientific Applications International, Inc., Oct. 1984.

[Bixler and Miller 87]
J. Patrick Bixler and David P. Miller.
A Sensory Input System for Autonomous Mobile Robots.
In *Proceedings of the Workshop on Spatial Reasoning and Multi-Sensor Fusion.* American Association for Artificial Intelligence, Morgan Kaufman Publishers, Inc., 1987.

[Blostein 87] Dorothea Blostein.
Recovering the Orientation of Textured Surfaces in Natural Scenes.
PhD thesis, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, April 1987.

[Blostein and Ahuja 87]
Dorothea Blostein and Narendra Ahuja.
Representation and Three-Dimensional Interpretation of Image Texture: An INtegrated Approach.
In *First International Conference on Computer Vision*, pages 444-449. IEEE, Computer Society Press, June, 1987.

[Brou 83] Philippe Brou.
Finding the Orientation of Objects in Vector Maps.
PhD thesis, Massachusetts Institute of Technology, July 1983.

[Brou 84] Philippe Brou.
Using The Gaussian Image to Find the Orientation of Objects.
The International Journal of Robotics Research 3(4):89 - 125, 1984.

[Brown 86] Lisa Gottesfeld Brown.
A parallel Implementation of Witkin's Surface From Texture Algorithm.
Technical Report, Columbia University, September 1986.

[Brown and Shvaytser 88]
Lisa Gottesfeld Brown and Haim Shvaytser.
Surface Orientation from Projective Foreshortening of Isotropic Texture Autocorrelation.
In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1988.

[Davis 82] Larry S. Davis.
Image Texture Analysis: Recent Developments.
Pattern Recognition and Image Processing :214-217, June 1982.

[Davis, Janos and Dunn 83]
Larry S. Davis, Lubvik Janos, and Stanley M. Dunn.
Efficient Recovery of Shape from Texture.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(5):485-492, September 1983.

[Dunn 84] Stanley M. Dunn, Larry S. Davis, and Hannu A. Hakalahti.
Experiments in Recovering Surface Orientation from Texture.
Technical Report CAR-TR-61, University of Maryland Center For Automation Research, May 1984.

[Dunn 86] Stanley M. Dunn.
Recovering the Orientation of Textured Surfaces.
Technical Report CAR-TR-193, University of Maryland Center For Automation Research, March 1986.

[Fekete 83] Gyorgy Fekete.
Generating Silhouettes of Polyhedra.
Technical Report CAR-TR-48, University of Maryland Center For Automation Research, December 1983.

[Fekete and Davis 84] Gyorgy Fekete and Larry S. Davis.
Property Spheres: A New Representation For 3-D Object Recognition.
Proceedings of the Workshop on Computer Vision Representation and Control :192 - 201, 1984.

[Gibson 50] James J. Gibson.
Perception of the Visual World.
Riverside Press, 1950.

[Hamey 88] Leonard G.C. Hamey.
Computer Perception of Repetitive Textures.
PhD thesis, C.M.U., 1988.

[Haralick 78] Robert M. Haralick.
Statistical and Structural Approaches to Texture.
Proceedings of the Internation Conference on Pattern Recognition :45-69, November 1978.

[Hillis 81] W. D. Hillis.
The Connection Machine.
Technical Report A. I. Memo 646, Massachusetts Institute Technology AI, 1981.

[Horn 70] B.K.P. Horn.
Shape from Shading: a Method for Obtaining the Shape of a Smooth Opaque Object from One View.
Technical Report MAC-TR-79 , Project MAC, MIT, 1970.

[Ikeuchi 80a] Katsushi Ikeuchi.
Shape from Regular Patterns (an Example from Constraint Propagation in Vision).
Proceedings of the Internation Conference on Pattern Recognition :1032-1039, December 1980.

[Ikeuchi 80b] Katsushi Ikeuchi.
Numerical Shape from Shading and Occluding Contours in a Single View.
Technical Report AI Memo 566, Massachusetts Institute of Technology, 1980.

[Ikeuchi 87] Katsushi Ikeuchi.
Determining a Depth Map Using a Dual Photometric Stereo.
Robotics Research 6(1):15-31, Spring 1987.

[Ikeuchi and Horn 81]
Katsushi Ikeuchi and Berthold K.P. Horn.
Numerical Shape from Shading and Occluding Boundaries.
Artificial Intelligence 17"(2):141-184, 1981.

[Jau and Chin 88] Y.C. Jau and Roland T. Chin.
Shape from Texture Using the Wigner Distribution.
In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1988.

[Kanatani and Chou 86]
Ken-ichi Kanatani and Tsai-Chia Chou.
Shape from Texture: General Principle.
In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1986.

[Kender 80] John R. Kender.
Shape from Texture.
PhD thesis, C.M.U., 1980.

[Kender 83] John R. Kender.
Surface Constraints from Linear Extents.
Proceedings of the National Conference on Artificial Intelligence , March 1983.

[Kender and Moerdler rt]
John R. Kender and Mark L. Moerdler.
Shape from Eccentricities Under Perspective.
Technical Report, Columbia University, Department of Computer Science, Forthcoming Technical Report.

[Kjell and Dyer 86] Bradley P. Kjell and Charles R. Dyer.
Segmentation of Textured Images.
Computer Vision Pattern Recognition :476 - 481, 1986.

[Korn and Dyer 86]
Matthew R. Korn and Charles R. Dyer.
3-D Multiview Object Representation for Model-Based Object Recognition.
Technical Report RC 11760, IBM T.J. Watson Research Center, March 1986.

[Laws 80]
Kenneth Ivan Laws.
Textured Image Segmentation.
PhD thesis, University of Southern California, Janurary 1980.

[Lee 87]
David Lee.
Algorithms for Shape from Shading and Occluding Boundaries.
Technical Report, Bell Laboratories, 1987.

[Lee, Pavlidis, and Huang 88a]
David Lee, Theo Pavlidis, and Kai Huang.
Edge Detection through Residual Analysis.
In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1988.

[Lee, Pavlidis, and Huang 88b]
Robert Haralick and S. J. Lee.
Context Dependent Edge Detection.
In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1988.

[Mackworth 73]
A. K. Mackworth.
Interpreting Pictures of Polyhedral Scence.
Artificial Intelligence 4(2):121-137, Summer 1973.

[Magee 84]
M.J. Magee and J.K. Aggarwal.
Determining Vanishing Points from Perspective Images.
Computer Vision, Graphics, and Image Processing 26:256-267, 1984.

[Marr and Hildreth 79]
D.C. Marr and E. Hildreth.
A Theory of Edge Detection.
Technical Report AI Memo 518, Massachusetts Institute of Technology, 1979.

[Marr and Poggio 79]
D.C. Marr and T. Poggio.
A Computational Theory Of Human Stereo Vision.
In *Proceedings Royal Society of London*, pages 301-328. Royal Society of London, 1979.

[Moerdler and Boult 88]
Mark L. Moerdler and Terrance E. Boult.
The Integration of Information from Stereo and Multiple Shape-From-Texture.
In *Computer Vision Pattern Recognition*. The Computer Society of the IEEE, The Computer Society Press, 1988.

[Moerdler and Kender 85]
Mark L. Moerdler and John R. Kender.
Surface Orientation and Segmentation from Perspective Views of Parallel-Line Textures.
Technical Report, Columbia University, 1985.

[Moerdler and Kender 87a]
Mark L. Moerdler and John R. Kender.
An Integrated System That Unifies Multiple Shape From Texture Algorithms.
In *Proceedings of AAAI 87*, pages 723 to 727. AAAI, Morgan Kaufman Publishers, Inc., 1987.

[Moerdler and Kender 87b]
Mark L. Moerdler and John R. Kender.
An Approach to the Fusion of Multiple Shape From Texture Algorithms.
In *Proceedings of the Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 272to281". Morgan Kaufman Publishers, Inc., 1987.

[Moerdler and Kender 87c]
Mark L. Moerdler and John R. Kender.
An Integrated System That Unifies Multiple Shape From Texture Algorithms.
In *Proceedings of the Image Understanding Workshop*, pages 574 to 580. DARPA, Morgan Kaufman Publishers, Inc., 1987.

[Ohta et. al. 81] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai.
Obtaining Suface Orientation from Texels under Perspective Projection.
In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. IJCAI, IJCAI, 1981.

[Pentland 86] Alex P. Pentland.
Shading into Texture.
Artificial Intelligence 29(2):147-170, August 1986.

[Raafat and Wong 86]
Hazem M. Raafat and Andrew K.C. Wong.
Texture-Based Image Segmentation.
Computer Vision Pattern Recognition :469-475, 1986.

[Serra 74] J. Serra.
Theoretical Bases of the Leitz Texture.
Leitz Scientific and Technical Information :125-136, April, 1974.

[Shafer, Kanade, and Kender 83]
Steven A. Shafer and Takeo Kanade and John R. Kender.
Gradient Space under Orthography and Perspective.
Computer Vision, Graphics and Image Processing (24):182-199, 1983.

[Spatial Reasoning and Multi-Sensor Fusion 87]
American Association for Artificial Intelligence.
Proceedings of the Workshop on Spatial Reasoning and Multi-Sensor Fusion, Morgan Kaufman Publishers, Inc., 1987.

[Stevens 79] Kent A. Stevens.
Surface Perception from Local Analysis of Texture and Contour.
PhD thesis, Massachusetts Institute of Technology, 1979.

[Terzopoulos 86] Demetri Terzopoulos.
Integrating Visual Information From Multiple Sources.
From Pixels to Predicates: Recent Advances in computational and Robotic Vision.
Ablex Publishing Corp., 1986, pages 111 - 142.

[Witkin 80] Andrew P. Witkin.
Recovering Surface Shape from Orientation and Texture.
Computer Vision.
North-Holland Publishing Company, 1980, pages 17-45.

[Wolff 86] Larry Wolff.
Parallel Algorithms for the Determination of Shape from Texture.
Technical Report, Columbia University, 1986.

Table of Contents