STEREO 3-D PERCEPTION FOR A ROBOT

Thesis by

Scott Darrell Roth

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1978

(Submitted March 17, 1978)

ACKNOWLEDGEMENTS

First, I thank God that the thesis process terminates. Second, I thank Ivan Sutherland, my advisor, for his patience with me and the process.

I am grateful to Caltech for the use of its facilities, for its financial assistance to me, and for its bright community of people. I am particularly indebted to my Caltech friends: Jeena Srinivas, Shriram Udupa, Mike Shantz, Mike Hyson, Dan Diner, Gilbert McCann, Giorgio Ingargiola, and Ron Ayres. I am also grateful to William Whitney and Len Friedman for their help, and for their exciting robotic's laboratory at JPL. Last of all, I thank Bela Julesz for his beautiful random-dot stereograms.

ABSTRACT

This thesis concludes a study of robot vision and presents an analysis of the rudimentary vision problem of modeling form in 3-Space. A stereo "snapshot" vision theory for a computer is proposed, based on an experimental implementation.

Here, stereo vision is argued to be essential for modeling natural or unfamiliar domains. The "firm" perception resulting from stereopsis is second only to kinesthesia/tactility in effectiveness with the unknown.

The novel mechanism introduced—as the heart of the system—is a stereopsis algorithm for growing stereo surfaces in natural scenes. First, 2-D features are extracted from the stereo pair of digital images by locating patterns of change in the images' "gradient-arrow" representations. Then, by associating features in the left image with features in the right image, stereo regions or "matches" are made. The stereopsis process fuses the stereo images by growing contexts of matched features. Every match defines via the camera geometry a visible surface in the scene, interlocking with neighboring matches like the pieces of a 3-D jigsaw puzzle. The resultant surface molds provide a firm basis for a polyhedral model of the scene's forms.

TABLE OF CONTENTS

I. INTRODUCTION

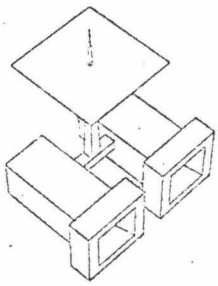
Television kills telephony in brothers' broil.
Our eyes demand their turn. Let them be seen!

James Joyce[7]

I A. SIGHTED ROBOTS

The goals of this thesis were formed within the framework of the JPL Robotics Research Program. NASA, funder of the program, is investigating the feasibility of planetary surface exploration by unmanned, semi-autonomous roving labs. An experimental robot vehicle with stereo cameras, laser rangefinder, multi-jointed arm, and touch/proximity sensors has been built and is in operation.

Putting aside space technology's special constraints such as weight, power, and reliability, one of NASA's major problems in building a mechanical astronaut is visual perception. In fact, development of machine vision systems is a major technical problem faced by the general robot industry. Consider NASA's requirements for sensory perception. The relevant environment of the robot must be sensed and modeled for the following activities:



- Setting-up and performing experiments
- Monitoring manipulation
- Navigation and guidance for the vehicle
- Locating "interesting" rocks, life forms, etc.
- Discovery of unexpected phenomena

The first three categories of activity, above, do not absolutely need a general purpose vision system. A scene understanding system could be specially designed to recognize the NASA-built world of the robot. Tools and mechanical parts can be designed for visual clarity and manipulation can be performed blindly. For vehicle guidance, a conservative strategy would be to make a slow laser scan off the bow, blindly creep a short distance, stop, make a slow laser scan off the bow, etc.

Vision methods for general domains are needed for robot activities in unfamiliar natural worlds.

I B. "FIRM" VISION PSYCHOLOGY

What are the rudimentary mechanisms of visual perception which must have greatest generality in order to perceive unexpected or new visual domains? What are the unit features—the atomic graphic symbols—of an image to be extracted?

The familiar world is rich with *soft vision* clues. The more familiar the scene, the less dynamic information is required by us to interpret its content.

Evidence: The well placed simplicity in cartoons.
The eye's selective scans of the scene.

Soft vision is semantically based perception. The perceptual process relies on making selective probes of the image's features to confirm an anticipated scene.

Soft vision clues are sought out and learned to reduce the perceptual effort required in familiar visual environments. For dynamic human activities, use of soft clues to minimize investigative movement and time is a necessity. Identifying the parts of a scene from a single viewpoint involves speculation about hidden surfaces. Self-occlusion alone ambiguates form by usually hiding at least 50% of surfaces—the back sides.

2-D images, from line drawings to photographs, test the high level soft vision that visual experience nurtures. This cultural perception has been extensively studied by vision psychologists.

> Piaget's "primary illusions" are 2-D figures, in response to which children make greater quantitative errors in perceptual judgment than adults[13]. By "primary", Piaget did not mean innate. The youngsters had been learning to read the language of their visual worlds for at least 5 or 6 years prior to testing.

Despite the speed and competence of mammals' vision, a furry creature has yet to find an Escher drawing of an impossible scene perplexing—or even interesting.

The *softness* of 2-D image understanding has been firmed by case studies of recovery from congenital blindness.

> An interesting example[3] dates back to 1959 when a successful corneal graft gave sight to a fifty-two year old man who had been blind since infancy. When first interviewed and tested by psychologists weeks after the operation, he demonstrated surprising competence at visually separating out and identifying objects in 3-Space, relating their visual form to his refined tactile models. When shown a picture of the famous Necker cube "illusion", a wire cube drawing, there was no depth preference or ambiguity experienced. In fact, he failed to understand it as a cube despite a motivating explanation by the experimenters accompanied by a real toy block for fondling and illustration. Hyper-objectivity was demonstrated with all of the 2-D illusions presented. "Apparent depth was not evoked by perspective drawings" was a conclusion of the case study.

Soft vision is powerful and efficient, but it falters with the unexpected and is useless with the completely new. Much of the visual space filled by nature and art—though not necessarily providing a completely new visual experience—is only form, with no semantic content. What are the visual features of a rock? Rocks may be sculpted into any shape and size. There may be no straight lines, much less a "preferred perspective viewpoint". Even in simple single-light-source environments, every variance in the reflectance properties of a rock's surface is a misleading illumination clue for shape.

The primitive visual mechanisms for modeling form in 3-Space seem to be more intimately related to kinesthetic perception and to be less symbolic than is popularly believed. Objects such as rocks have no intrinsic value other than filling kinesthetic/tactile space. Locating

the surfaces of obstacles to manipulation and locomotion does not involve the more difficult task of identifying them.

Kinesthetic/tactile senses are *hard*. There are no kinesthetic figure/ground ambiguities, no effective kinesthetic camouflage.

The *firm* depth clues of stereopsis and, to a somewhat less extent, motion parallax are second only to kinesthesia/tactility in effectiveness with the unknown. Locating surfaces in 3-Space with stereopsis does not require 2-D image understanding. Julesz has demonstrated this by synthesizing the "2-D unknown" with random-dot stereograms[8]. (A random-dot stereogram is a plane of texture monocularly, but a vivid illusion of form in 3-Space when binocularly fused). Although motion parallax is similar in principle to stereopsis, two differences should be noted. First, variable head movement must be precisely measured whereas the fixed distance between eyes is "known". Second, dynamic scene changes over time must be correlated whereas the action is frozen by the "simul-snapshot" of stereo vision.

The stereopsis process *objectively* establishes surfaces in 3-Space by fusing two "syntactic mosaics" representing the stereo images from a pair of cameras. The mosaic tiles—varying in size, shape, brightness, etc.—are the fusable elements, features defined by making distinctions within each 2-D camera image. These features merely fragment the images into areas. Before fusion, they are uncommitted to perceptual objects or even to neighboring features. "Merging", "dissolving weak boundaries", and "subregions" are not working concepts before stereo fusion. When an area in one camera image is matched with an area in the other, a surface in 3-Space is defined via the camera geometry. And although neighboring visible surfaces in 3-Space will correspond to

neighboring features in the 2-D images, the converse is not necessarily true.

Semantic scene understanding systems which are non-interactive and include no firm depth clues have succeeded only in small domains. The high level vision systems developed for dealing with block worlds[18], road scenes[19], and office environments[1] succeed in their special domains by using expectations. High level vision research is important. However, I think stereopsis should be developed before dealing with the more context dependent information in soft clues.

In a stereo vision system, as in kinesthesia, perceptual reasoning begins in 3-Space, not in the image. The first perceptual action following fusion may be to simplify the representation of the scene by merging smoothly contiguous surfaces, making partial molds of the scene's major forms. Then, by applying a principle of symmetry, occluded surfaces—particularly back sides—may be hypothesized, thereby completing the molds. Finally, by applying semantic analysis to the forms in familiar environments, objects may be identified.

For modeling form in 3-Space, stereo vision makes the classic pattern recognition problem irrelevant. The following quote raises the central pattern recognition issue.

> "How can local characteristics of texture elements ... be globally bound together into a perceptual region?" [20]

For most examples of patterns, from tree bark to lawns, the answer to the question is "The surface binds the elements!". The texture elements are the fusable features of the image and the product of the fusion process is surfaces, from tree trunk cylinders to ground planes.

A pattern recognition problem that is not solved by simple stereo surface-bindings is textured volume. A sparsely foliaged tree and a flock of butterflies are both perceptual objects whose parts are disjointly distributed in a volume. The problem is to capture the volume by grouping the visible parts into a perceptual whole for identifying it and maybe for simplifying its representation. When viewed from afar, there is no problem because the volume of foliage lies in one or two binocular planes, the binding surfaces. But when viewed closely or from within, the elements of the foliage retain their individuality. Grouping the elementary forms by tracing stem connections in a tree may be possible, but it still leaves butterflies unflocked. Clearly, this is a symbolic perceptual problem dealing with "proximity" and "sameness" relations, and not a rudimentary vision task.

The important extension to a stereo "snapshot" vision theory is a movie processing capability. The practical implications are more significant than the ability to track animate objects. The *redundancy of form* in stereo movies should make robot vision economical for wide usage. Continuous movement of sensors causes continuous movement of features in the scene's image. Consequently, each frame of a movie is only partially new—depending on the relative speeds of movement and image processing, of course. It should be possible to anticipate most of the changes in stereo images caused by small steps in the stereo cameras' movement.

I C. MACHINE VISION LABORATORY

Before jumping into a summary of the vision system, I describe the environment for the implementation. Caltech's PDP-10 by DEC has been available without usage limitations other than the built-in constraints of an old, heavily used, time-shared computer. The high-level programming language SAIL, Stanford Artificial Intelligence Language[14], was used almost exclusively.

The stereo images used in this research were made using a single camera, A/D converter, and PDP-11. Stereo pairs were synthesized by shifting the scene, an easy but limited method. Precision in shifting the scene and modeling the camera was not attempted, despite non-linear distortions proved by an image of straight graph paper. Precise measurements and alignments of cameras would be important in a production system, but were unnecessary for this research. Camera calibration methods are well documented elsewhere.

Each digital image is a square array of about 500x500 picture elements (pixels). Each pixel is an 8-bit number, range 0-255, representing a logarithmic light intensity value. The actual correspondence of these pixel values to luminance is unknown.

Digital image processing is expensive, in both time and space. On Caltech's PDP-10, a pair of digital images packed 4 pixels/word requires 125K words of memory. But maximum job space is only 56K, and that includes both data and code. Consequently, the dense images were stored on disk and only windows of the image, about 100 x 100 pixels, were used. Usually the window encompassed more than 100x100 original pixels; square areas of pixels from the original image were averaged to produce

new pixels for the window. For instance, when the entire original image was used, a 5x5 square group of pixels was averaged to produce one pixel in the 100x100 window.

The machine environment, single processor and 56K of "core" backed up by disk, influenced the design of the vision system. The two relevant hardware variables are the quantity of memory and number of interconnected processors—the number of processors being most important. If many processors had been available, the software design concepts might have been radically different.

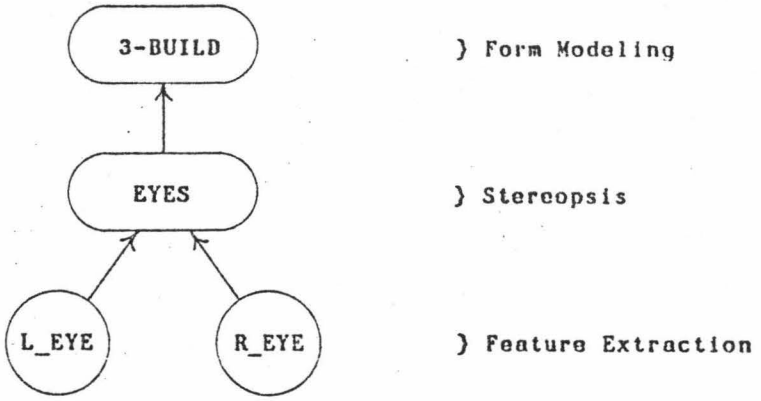
How much do the CPU and memory demands grow with increasing image density? This is the critical inquiry into the extensibility of vision systems which use low density images. For the system that this thesis documents, CPU and memory usages grow linearly with the image area if maximum allowable stereo disparity is fixed. But if the fusable disparity range grows with the width of the image, then processing time for some stereopsis functions would increase proportionally to $(A^{3/2}+A)/2$ where A is the area of the square image. Memory usage would still grow at the linear rate A, however. Specific CPU and memory costs will vary, of course, depending on the complexity of the stereo images. See V. Conclusions for an analysis of image complexity and estimates of the CPU costs.

The utility programs I wrote for the PDP-10 to assist in implementing the vision system are not described in the body of this thesis but are worth noting. I provided device plotting services from SAIL, a 3-D graphics system, image file handling, etc. A Hewlett-Packard plotter, Tektronix scopes, and a Diablo printer with plot mode were used to display results at various process stages. This

capability was crucial for debugging and, secondarily, provided some illustrations for this thesis.

I D. SYSTEM OVERVIEW

The vision system implemented may be simply described as a computer program whose input is a stereo pair of digital images and whose output is a polygonal surface model of the scene. The "computer program" implements a three step process: feature extraction, stereopsis, and form modeling. The three steps are described—in this bottom-up order—in chapters II, III, and IV of this thesis.



Subprocess hierarchy

Four computer modules (or jobs) with acronyms L_EYE, R_EYE, EYES, and 3-BUILD are the subprocesses. L_EYE and R_EYE execute independently, so together are a single process step. Digital images are given to L_EYE and R_EYE; polygons are generated by 3-BUILD.

Stereo images and a polygonal surface model are both representations of the visual scene, the beginning and end of a data structure history.

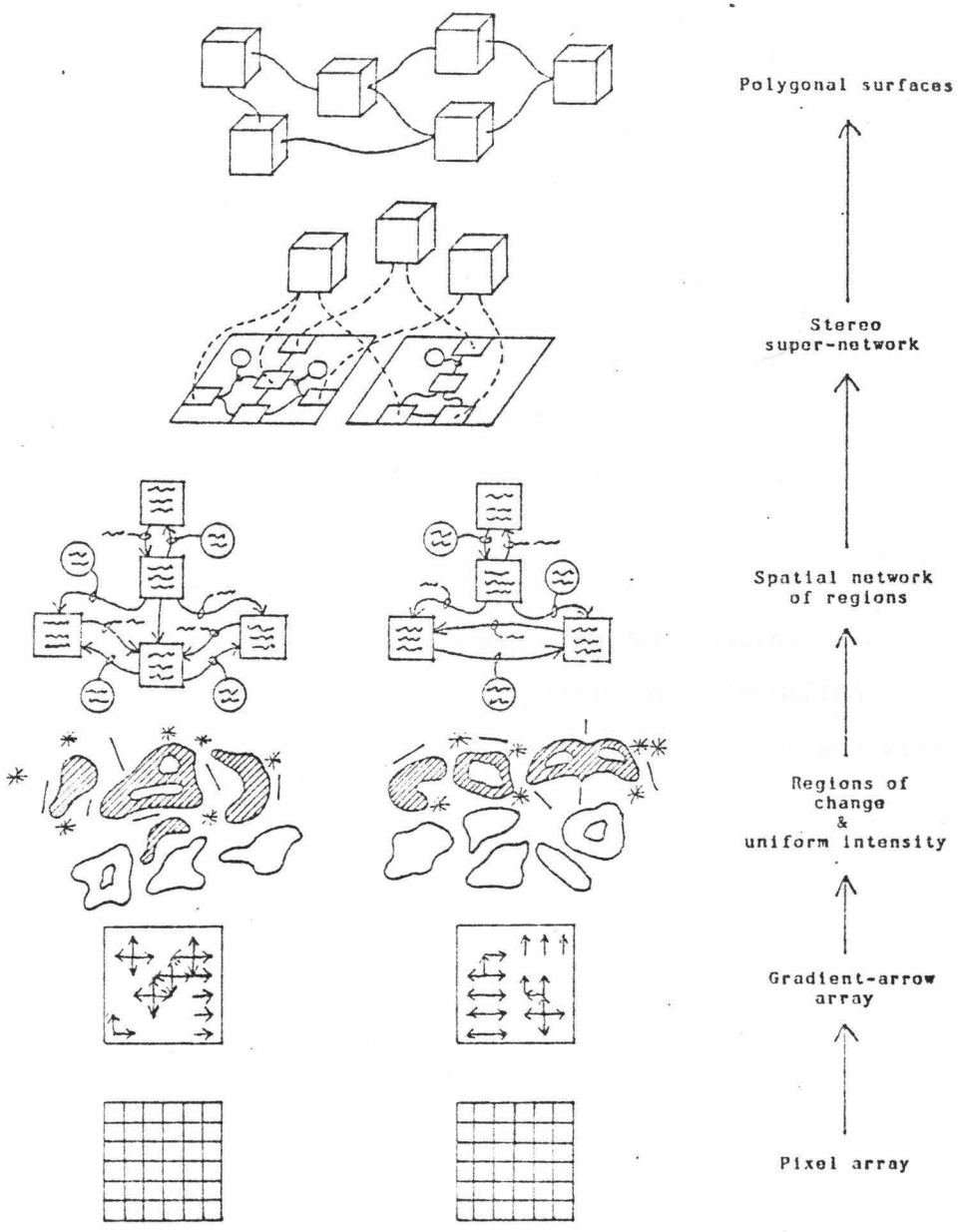


Image to form transformation

The first operation on the pair of digital images, at the bottom, is to locate all *gradient-arrows*. A gradient-arrow is the primitive distinction in the image array, representing a difference in intensity between two neighboring pixels greater than some threshold of

sensitivity. By convention, the picture of a gradient-arrow points from the pixel of brighter intensity to the lesser one. The gradient-arrow is merely a directed intensity gradient, an edge between two pixels. The new term is introduced here because it denotes a powerful graphic symbol that clearly depicts patterns of intensity change in the image. By representing the gradient arrows in a binary array, the program can quickly extract features from the images.

From the gradient-arrow arrays and intensity arrays, regions are defined. The regions are the features of the image, filling the entire array. Arrowless regions are areas of uniform intensity and regions with arrows are *patterns of change*. Typical patterns of change are edges, bars, shade, blobs, etc.—to use feature mask terms[10]. Unlike masks, however, the patterns are regions, varying in shape and size.

After the regions have been formed, they are characterized and arranged into a spatial network. The network's spatial relations connect neighboring regions. There are two types of neighbor relations: "inside of" and "outside of". The region characterizations are sufficiently complete so that the pixel and gradient-arrow arrays may be discarded. Indeed, at this point of the process, control passes from L_EYE and R_EYE to EYES; and the two spatial networks of regions are the data structures passed upward.

EYES is the stereopsis process that fuses the stereo pair of images. Every region(s) to region(s) match defines a stereo region called a *steregion*. The fusion task is to build a *super-network of steregions*, connecting the two spatial networks, region by region. This network construction is not a direct computation. Rather, the control structure is like a relaxation process for solving 3-D jigsaw puzzles.

Every steregion defines a surface in the visual 3-Space, usually interlocking with surfaces defined by neighboring steregions. Multiple independent contexts of steregions in the network—and, hence, in the visual space—may be under development concurrently. When contexts grow to become neighbors in the images, they become dependent. The result is conflict or confirmation. Matches are made and broken as the stereopsis process converges to a stable state of fusion.

Last of all, form in 3-Space is modeled by polygonal surfaces—that is, by partial polyhedra. A match of a point in one image with a point in the other image defines, via the camera geometry, a point in the visible 3-Space. It follows that a stereo pair of region boundaries defines a surface outline in 3-Space. Most outlines may be simply covered by polygons. If, however, a matched region has other regions inside of it, the steregion's "surface" may be a hole or a mound. In this case, the interior steregions define the spatial contour within the outer steregion's boundary.

I decided to use polyhedra to represent form for three reasons: 1) they are sufficiently general—using very small polygons, if necessary—for all shapes, 2) there is an old graphics tradition of using polyhedra, and 3) the cameras' geometric model consists only of straight lines. True, a polyhedron representation of a sphere is not very practical. Any single shape formulation, such as generalized cones or cylinders, will describe certain shapes inefficiently, particularly if available resolution of detail in form is to be retained. The important result is that the visual form is well-defined at this terminal stage of the process.

Are partial polyhedra sufficient or should all visible surfaces be boundaries to specific, completely enclosed volumes? The system implemented produces only polygonal molds for the visible surfaces in the scene. Completing the molds would produce polyhedra. Many surfaces such as the ground, enclosing walls, and the features on the farthest depth plane (e.g., mountains and sky) do not need voluminal description. However, the system should assume that objects to be handled or obstacles to be avoided do occupy a volume. For primitive needs, a rough model of the occluded parts would be sufficient. Robot vehicle and arm navigation systems have been designed to work with such domain uncertainties[15,17]. Further, kinesthetic/tactile perception may be used to complete the object's visual model and dynamically alter planned action[2].

Without kinesthetic/tactile exploration and investigative movement to other viewpoints, completion of the partial polyhedra requires speculation about the occluded parts of the scene. A simple, crude method of speculation would be to fabricate back sides by merely truncating the occluded volume (in the rear). The "objects" modeled would be contiguous polygon groups and the depth of an object may be assumed to be a diameter of the visible front. For refinement, heuristics of symmetry based on centroids or major axes could be used by a form analysis process.

II. FEATURE EXTRACTION

II A. OBSERVATIONS

The effect of the feature extraction process is to reduce the dense array of light intensity values to a tractable data structure for the fusion process. The data structure's elements are features, defined by making distinctions in the image. The features are interrelated spatially, so the representation of the image is a spatial network of features.

Most of the many feature extractors developed in the field of picture processing are of two general types: region growers and edge locators. To the region grower, visual distinctions are between neighboring features; but to the edge locator, the distinctions are the features.

Region growers segment a picture into features by merging neighboring pixels of similar intensity (examples: Ohlander[12] and Yakimovsky[19]). The range of intensities to accept is decided by analyzing intensity histograms for incidence discontinuities, which are assumed to delimit the intensity range of each "object" in the scene. Since histogram analysis operates outside of the the spatial domain, there is a danger of missing distinctions.

Edge locators filter out the intensity changes—leaving the area in the image uncollected—by using masks in various orientations and sizes (Marr[10]) or by using algorithmic operators which "fit" edges in circular neighborhoods of pixels (Hueckel[6]). Marr actually uses an assortment of masks, extracting bars (i.e., lines) and shade in addition to edges.

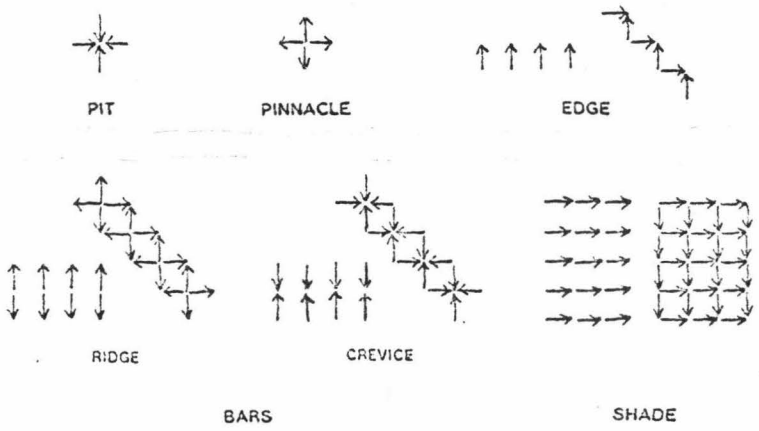
The feature extraction method I introduce here integrates the edge locator with the region grower. The flexibility and uniqueness of the open form of grown regions have been combined with the edge locator's "fixed focus" on the visual distinction. The features extracted are regions of uniform intensity and regions of change, formed in an intensity gradient representation of the image.

II B. GRADIENT-ARROWS

The most primitive distinction in an image is symbolized by the *gradient-arrow*, a directed edge that is one pixel in length. The image resolution prevents further refinement.

Gradient-arrows can be located in an image and graphically depicted by placing an arrow across every pixel edge which separates two pixels whose difference in intensity is greater than some threshold of perceptual sensitivity. By convention, the arrow points from the brighter pixel to that of lesser intensity, symbolic of a "greater than" relation.

The following figure uses gradient-arrows to illustrate the features that are often candidates for features masks.

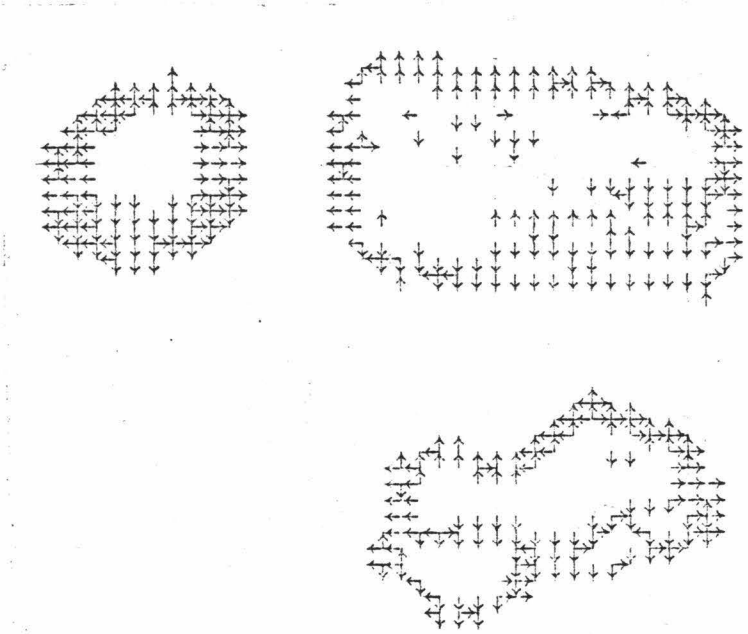


Well-behaved gradient-arrow patterns

These patterns—except the edge—characterize, albeit ideally, those found in images of natural scenes. A pixel, I assume, is a light intensity measurement not for a point but for an area, specifically for

a square because of the square arrangement of pixels. A consequence of the area measure is that edges will usually split a pixel, not merely fall between pixels. (With a hexagonal arrangement, all images of "straight" edges that are at least as long as a pixel will intersect a pixel). Thus, the gradient-arrow pattern for an edge is usually a line of shade, such as a column of double arrows pointing westwards.

In the following example, the observer should have no trouble interpreting different patterns of arrows.



Two rocks and one egg

A higher resolution image of an object such as an egg would not have such perfect edge and shade gradient-arrow patterns. But the fine imperfections caused by the variances of form, surface materials' reflectance, and lighting are features in themselves. Their recognition permits more accurate description of form and, in this "object" example, more discriminating identification.

II C. PATTERNS OF CHANGE

Fragmenting the image into features is a three step process.

- compute and encode the gradient-arrows
- group arrowless cells into regions of uniform intensity
- group gradient-arrow cells into regions of change

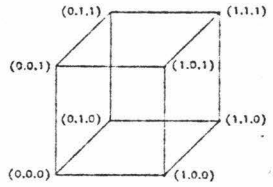
Forming a region of uniform intensity is a simple procedure: group neighboring arrowless cells and then extend the group's boundary to its enclosing edge. Specifically, the boundaries are extended outward one cell by "stepping into" arrow cells which have an arrow on the opposing edge.

Group Don't group

Arrowless cell extension examples

There are conflicting demands on the design of the strategy for grouping arrow cells into patterns of change. The strategy should reduce fragmentation by gathering across smooth transitions both in a pattern's orientation (e.g., the direction of shade about an egg) and in the intensity of gradients within a pattern (again, as in an egg's shade unit). Further, trendless gradients that are parts of features near the threshold of sensitivity or whose size is at or below resolution should be consumed into the major trends of their respective areas in the image. Consumption must be constrained, however, in order to preserve the separate identities of features.

Following non-exhaustive case analysis of arrow cell pairs and experimentation, I converged on a simple, heuristic grouping algorithm. The heuristic component of the algorithm's "merge conditional" is based on the simple *Hamming* Metric from communication theory. The *Hamming* Metric is a digital "difference" or "distance" measure between words of bits. The difference between two words is the number of bits that are different when the two words are compared on a 1-to-1 bit basis. The following figure is a drawing of a unit cube in 3-Space. There are 8 elements in the metric space of 3-bit words and they are the vertices of the cube. Further, each edge of the cube links 2 words whose *Hamming* difference is 1 and <u>all</u> differences of 1 are linked.



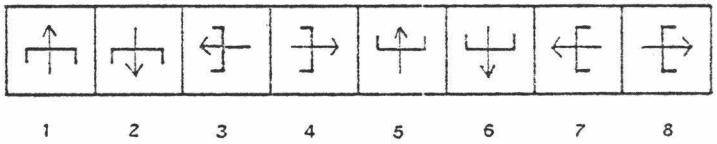
Unit *Hamming* difference cube

This same difference of 1, when applied to the binary representation of an arrow cell, forms the basis of the grouping rule.

An arrow cell is a square and each edge has three possible states:

- arrow pointing out
- arrow pointing in
- no arrow

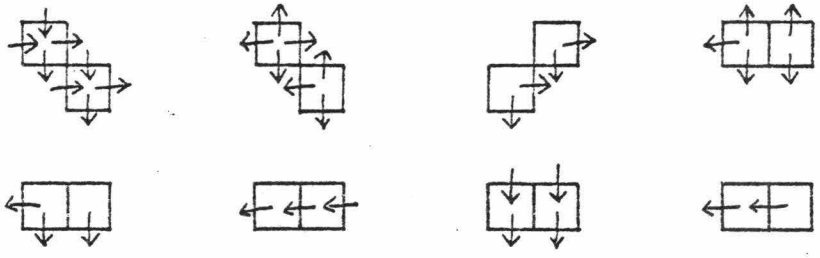
Consequently, there are 3^4 possible states for an arrow cell. Rather than using the minimum number of bits, 7, to encode the states, the square's 4 edges are separated by allowing each edge 2 bits to represent its 3 states.



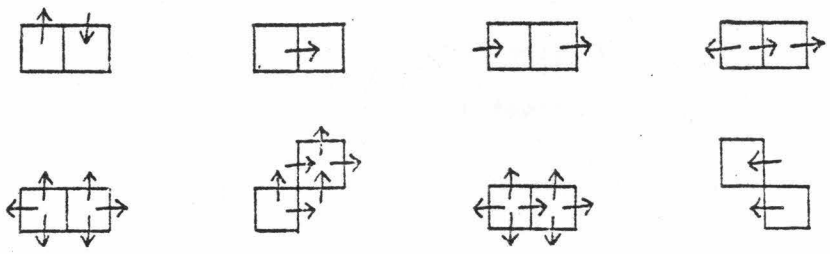
8-bit arrow cell

Note that ((B_1 AND B_2) OR (B_3 AND B_4) OR (B_5 AND B_6) OR (B_7 AND B_8)) is never true, where B_i means "bit i is ON".

The basic criterion of the grouping operation is to merge two arrow cells when their *Hamming* difference is 0 or 1. When the *Hamming* difference between two cell encodings is 0, they are identical. When the difference is 1, exactly one of the arrows which is present in one cell is missing from the other. That is, there is exactly one bit ON in a word that is not ON in the other.

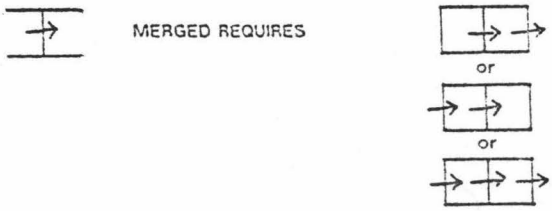


Examples: Merge



Examples: No merge

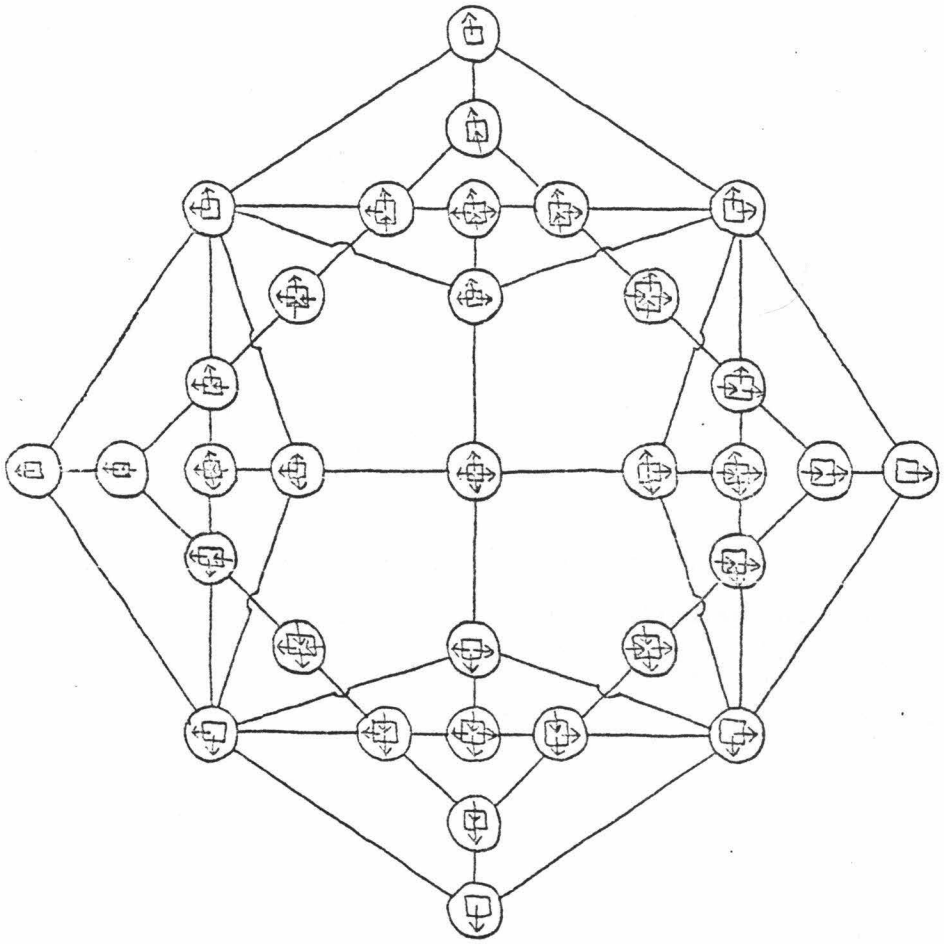
From the arrow cell's point of view, this means that arrow swaps are not allowed. That is, if the top edge state of one cell is "arrow pointing out", it would not be grouped with a neighboring cell whose top edge was in state "arrow pointing in". A greater consequence of the rule is that if there is any arrow ON between two cells, such as "greater right" in A and "lesser left" in B, an arrow in the same direction must be ON A's left edge or B's right edge to qualify for grouping.



Rule for grouping across a gradient, in effect

This means that arrow cells are grouped across an edge only if they are part of a trend of parallel edges at least 3 or 4 cells in length. This is the fundamental characteristic of shade.

The following picture shows most of those arrow cells that would be generated by an image of a sphere or egg. As in the earlier illustration of "Unit *Hamming* difference cube" on page 21, arrow cells with *Hamming* differences of 1 are linked. Note, however, that since horizontally and vertically neighboring cells share an edge, not all position combinations for linked pairs are possible. For instance, the center cell of the network below could only be a neighbor to its duplicate if diagonally arranged. See Appendix 1 for an accounting of all cells.



Group links for sphere arrow cells

The *Hamming* difference grouping criterion is only the basis of unit pattern determination. Even if two adjacent arrow cells pass this test, they may still fail to be grouped. There are two additional contraints.

1. If the adjacent cells are part of a shade trend—that is, they have an arrow between them—the intensities of the gradients in the trend must be examined to insure that the trend is smooth. As implemented, "smoothness" is the condition that the gradient intensity changes nearly linearly. The computation involves a line of 4 pixels of which the center 2,

separated by the center arrow, are in question. Where D_1, D_2, D_3 are the pixel intensity differences P_1-P_2, P_2-P_3, P_3-P_4 respectively, then the condition is that

$$\begin{aligned} &(\quad (D_1 > 0 \ \text{AND} \ D_2 > 0 \ \text{AND} \ D_3 > 0) \\ &\text{OR} \ (D_1 < 0 \ \text{AND} \ D_2 < 0 \ \text{AND} \ D_3 < 0)) \\ &\text{AND} \ (\ | \ (D_1 + D_3)/2 - D_2 \ | \ < \text{CONTRAST}) \end{aligned}$$

where CONTRAST is the original arrow threshold. If P_1 or P_4 is a member of a uniform intensity region, this condition is not enforced.

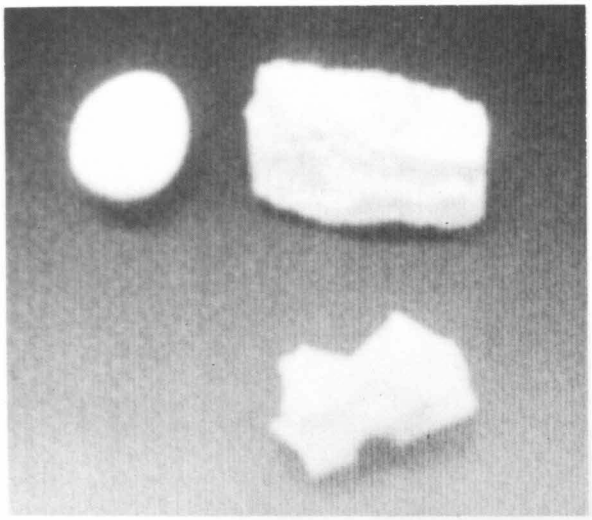
2. Special considerations are given to the grouping of diagonal cells. First, diagonal lines are weak, allowing intersecting areas of cells. Although the challenge of the game of Go exists because of this possibility, it is best eliminated here.

X	X	X	X	X
X	X	O	X	X
O	O	X	O	O
O	O	O	O	O

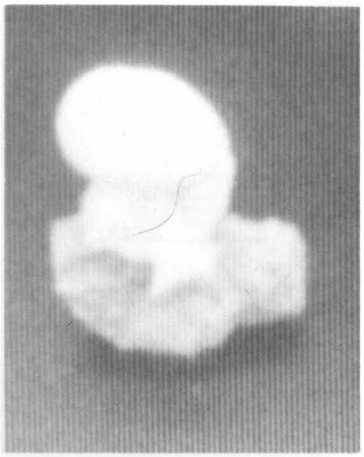
Regions X and O intersect each other

Second, diagonal cell pairs share only a point and are separated by a distance $2^{1/2}$ times greater than horizontal and vertical cells which share an edge. This muddles the geometry of the cellular image space. All of these problems would disappear, greatly simplifying the computer code for feature extraction, if cameras with hexagonal arrangements of light sensors were available.

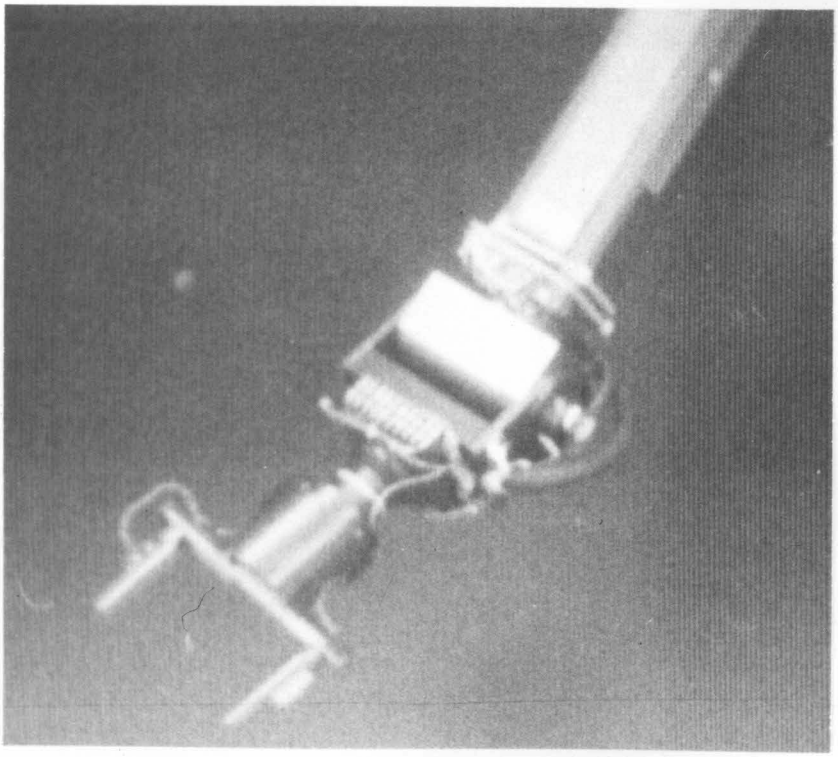
The following page contains three sample digital images. The reproduction here is poor because it is the printer's photograph of my photograph of the robot TV monitors. On the succeeding three pages, the gradient-arrow representations of the sample images are illustrated and the region boundaries of extracted features are plotted. Plus signs (i.e., +'s) indicate single cell regions, which I call "point regions".



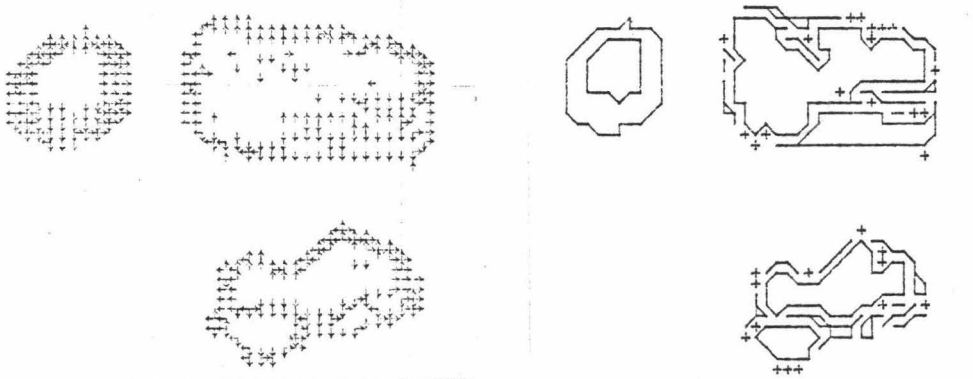
Two rocks and one egg



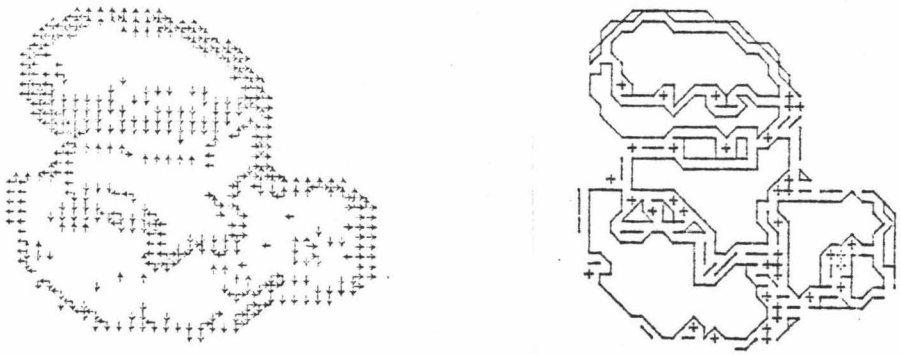
Egg and rocks stacked



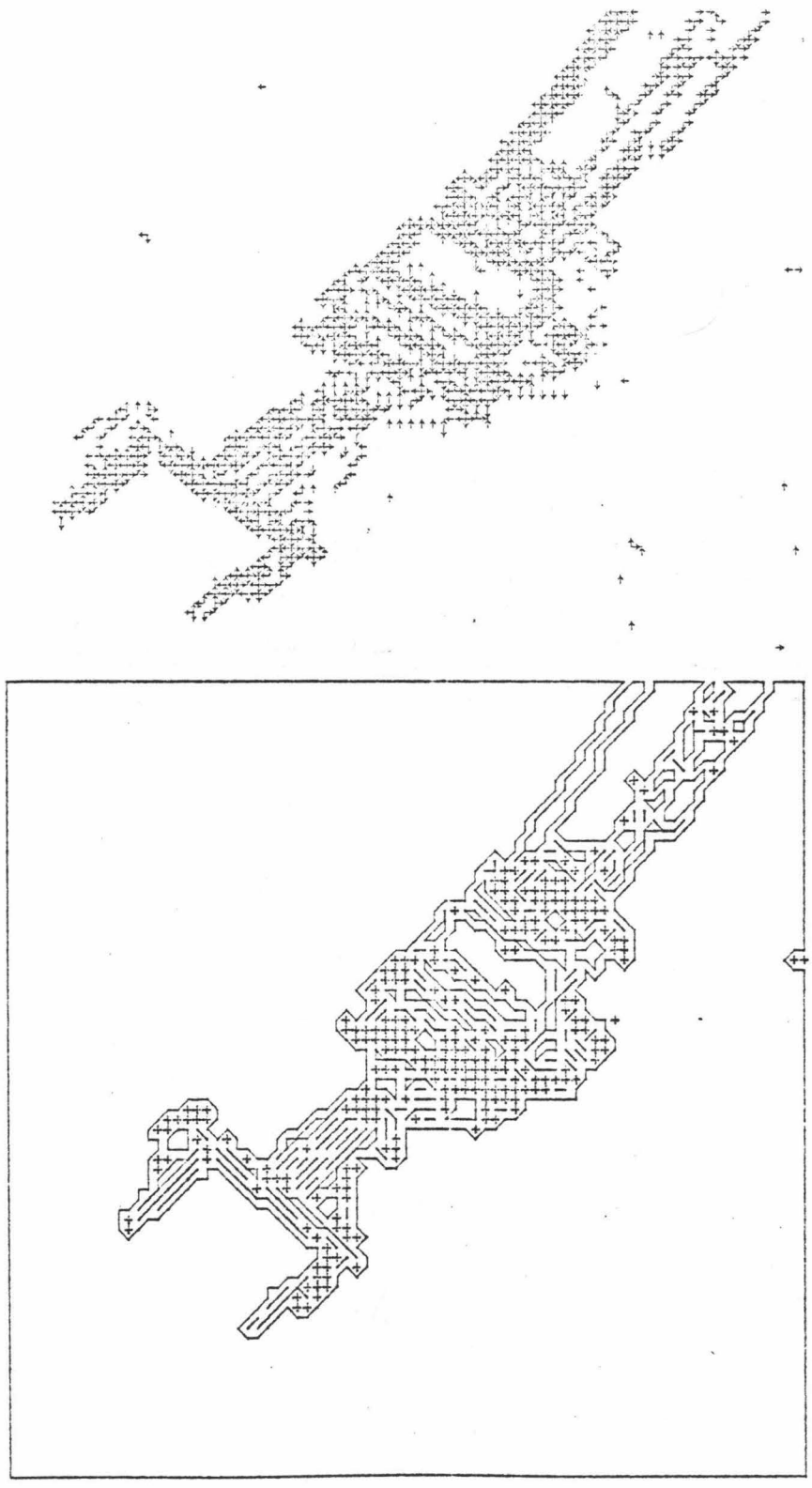
The JPL robot arm



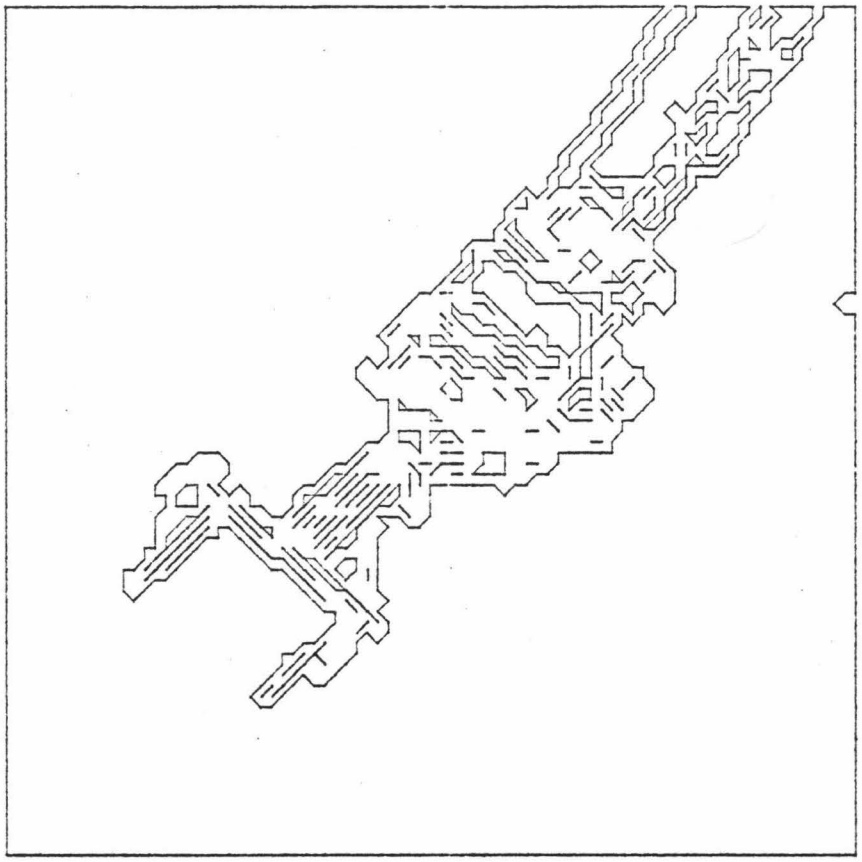
Two rocks and one egg



Egg and rocks stacked



The JPL robot arm



"Pointless" region boundaries

II D. UNIT PATTERN CHARACTERIZATION

After a set of contiguous arrow cells have been grouped to form a feature unit, they are characterized. Characterization of a feature unit produces a record data structure, called *REGION*, with the following subfields:

TYPE
LOCATION, BOTTOM, AREA, BOUNDARY_TRACE
INTENSITY, SHADE_VECTOR
<INSIDERS>, <OUTSIDERS>
<STEREGION>
TOUCHED_TIME

TYPE classifies REGION to be a uniform intensity region, a shaded region, or a point region. A shaded region is a region of change which has internal arrows—that is, arrows other than those along the region's boundary. The purpose of TYPE is merely to economize storage. If REGION is a uniform intensity region, then SHADE_VECTOR is unnecessary; if REGION is a point region, then BOTTOM, AREA, BOUNDARY_TRACE, SHADE_VECTOR, and INSIDERS are unnecessary.

LOCATION specifies the upper-most, left-most pixel of the region in the image. In the implementation, each LOCATION uses one word of storage: the line number in the left half and the column number in the right half.

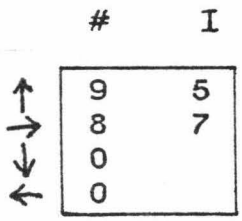
BOTTOM is the number of the last line of the image that contains the region.

AREA is the number of cells in the feature's region. The primary use of this information is to weight INTENSITYs when computing the average pixel intensity within two or more regions.

BOUNDARY_TRACE is a chain encoding of the region's boundary. The chain encoding is a string of 3-bit bytes each of which specifies one of 8 directions: north, south, east, west, and the 4 diagonals. Sequential execution of "move-to's" on the byte string would move the executor about the perimeter of the region, starting and ending at LOCATION.

INTENSITY is the feature's average intensity.

SHADE_VECTOR characterizes the trend of intensity change within shade. The vector is simply a table, listing for the four arrow directions their incidence and intensity within the region, boundary excluded. The following example describes an area of shade in the northeast direction. The average (gradient) intensity of the 17 arrows is 6.



Sample shade vector

SHADE_VECTOR is a summary, so some information about the distribution of a region's arrows is lost. Although this incompleteness in the characterization of shade seems to be insignificant, there is a possible alternative: eliminate INTENSITY and SHADE_VECTOR as REGION attributes

and have L_EYE and R_EYE pass their intensity arrays to EYES, the fusion process. Although more memory would be needed by EYES, there would be little additional CPU use. SHADE_VECTOR is referenced by EYES only during match evaluation, at which time the BOUNDARY_TRACEs of the match's regions are scanned in order to decompose the regions' 2-D forms into stacks of line intervals. At this same time the regions' intensity image could easily be scanned on a line by line basis for stereo correlation of intensity.

<INSIDERS> and *<OUTSIDERS>* are lists of pointers to other REGIONs. These are the neighbor relations that form the spatial network. The following section, II E, will explain.

<STEREGION>, initially null, is used in stereopsis as a pointer to a stereo region. See section III D.

TOUCHED_TIME is an integer variable used during the fusion process. See section III E.

II E. SPATIAL NETWORK

Following feature extraction, a structure is imposed on the feature characterizations, relating features in spatial proximity. Two binary relations, *INSIDE* and *OUTSIDE*, are the links. A feature's region R_a is OUTSIDE region R_b if 1) R_a is not enclosed by R_b and 2) at least one cell of R_a is adjacent to at least one cell of R_b. Region R_a is INSIDE region R_b if R_b is OUTSIDE R_a and R_a is not OUTSIDE R_b. This is consistent with the intuitive notions of inside and outside as they apply to touching pairs of contiguous areas in discreet 2-Space.

The OUTSIDE relation includes a description of the interface between the two regions. Specifically, substrings of a region's boundary trace that touch a neighbor's region are recorded and made available via pointers into the region's copy of its trace. If R_a is INSIDE of R_b, then this touching information is available in R_b's OUTSIDE relation with R_a.

The following illustrates a simple image's spatial network of features.

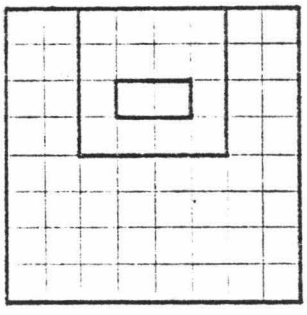
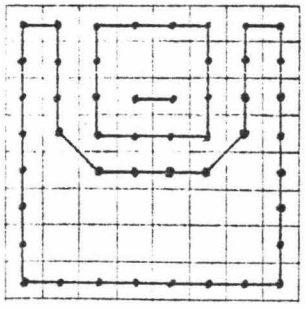


Image fragmentation



Region chains

The spatial relations *inside* and *outside* are the two neighbor relations implemented, but a third one is needed to complete the neighbor concept for the fusion process. This new relation may be thought of as being *nearest*. It relates *insiders* to nearby region boundaries. As the next chapter explains, stereo information is produced by fusing the boundaries of the match's regions. The "fit" of a match in its context is evaluated by comparing the fused positions of these boundaries in 3-Space with the fused positions of neighbors' boundaries. When a non-touching but close pair of regions are insiders of a larger region, for example, they are not directly related in the network although they are neighbors—two features which should influence each other's match, thus two features which should be readily accessible to each other.

Appendix 2 explains the problem and presents a solution. Adding *nearest* relations to the spatial network should involve little additional memory, CPU use, or even complexity. However, it would involve rewriting and debugging EYES which uses stereo pairs of the networks extensively.

III. STEREOPSIS

The following analysis of the stereopsis process and presentation of its mechanistic model are the major contributions of this thesis. I introduce the problem by explaining in section B the stereo camera geometry and by qualitatively describing in section C the possible and expected perspective differences between stereo views. Then I describe the fusion process in two parts: feature match evaluation in section D and the fusion process control structure in section E.

III A. OBSERVATIONS

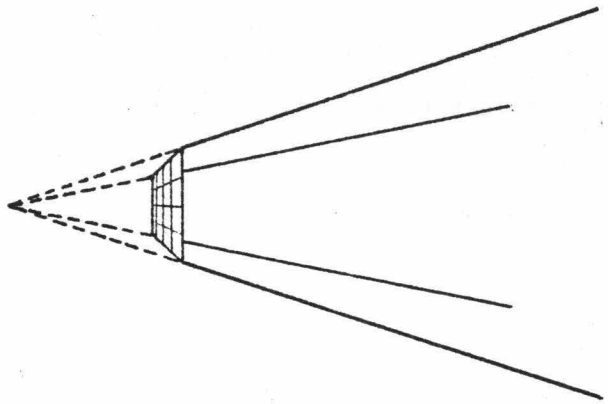
Hannah[4] implemented statistical "techniques for efficiently matching corresponding areas of a stereo pair of images". With a model of the stereo geometry, Hannah's system successfully projected some stereo image points into 3-Space.

Inspired by Julesz[8], Marr and Poggio[9] found a cooperative *algorithm* for fusing random-dot stereograms. The "scenes" were composed of regular planer shapes perpendicular to the viewer and the features were the dots themselves. Although primitive, it captured the spirit of the following important idea:

> depth is continuous over a greater area in images of natural scenes than it is discontinuous, so features influence the matching of neighboring features with the goal of minimizing discontinuities in the scene's apparent form.

III B. GEOMETRY OF BINOCULARITY

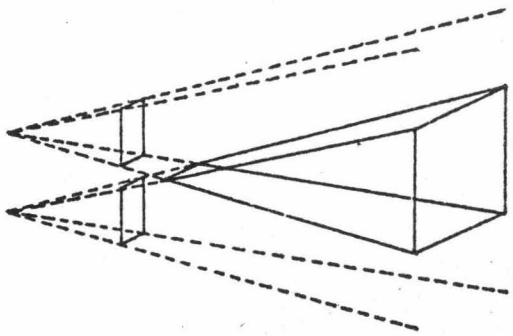
The ideal camera system modeled here has a thin lens and a sufficiently large depth of focus for the intended visual application. This system can be reduced to a simple 3-D geometry composed of a focal point, a square pixel array, and straight light rays which pass through the array to connect the focal point with the reflecting environment, one ray per pixel. The visual space is in the shape of a truncated four-sided pyramid.



Ideal camera system

The measurements of the rays' intensities are stored as pixels. The light source(s) or reflecting surface(s) responsible for a pixel's value intersect the pixel's ray, somewhere. Assuming a pixel is an area measure of light intensity, the rays are modeled as truncated pyramids, the packed bunch of which is the visual space.

For binocularity, two cameras are placed so that the image squares lie in the same plane and their edges are parallel. Of course, the focal lengths and the pixel resolutions and arrangements are the same. The binocular space is wedge shaped.



Ideal stereo camera system (top view, slightly rotated)

The wedge of binocularity is most usefully described as a "feather of isosceles triangles" each of which lies in a plane that contains both focal points. Each triangle intersects a single horizontal line in both pixel arrays. Consequently, *if a reflecting surface is visible to both cameras, it lies on the same horizontal line in both pixel arrays, ray by ray*. Conversely, if a pixel in one image is matched with a pixel in the other, the two rays they represent intersect at a point in the visual 3-Space—or are parallel and "intersect" at a distance of *pseudo-infinity*, as determined by the system's resolution.

Disparity, the amount of horizontal shift, is conveniently measured in pixels. The range of disparity is then 0 .. W-1 where W is the number of pixels in the width of the image array. If the disparity of two matched pixels is 0, their rays intersect on the back of the wedge at pseudo-infinity. If the disparity is W-1, the intersection point lies on the front edge of the wedge. Corresponding to the W values of disparity are W planes parallel to the images' plane. All possible points of intersecting rays with disparity *n* lie on plane *n*.

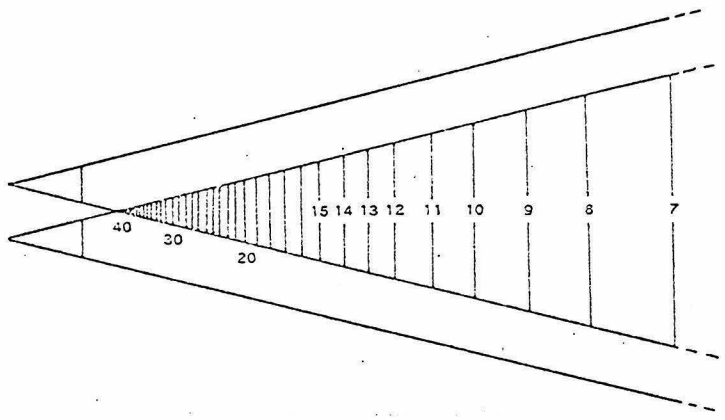


Image width = 1
41 pixels / image width
Focal length = 2
Separation between focal points = 1.5

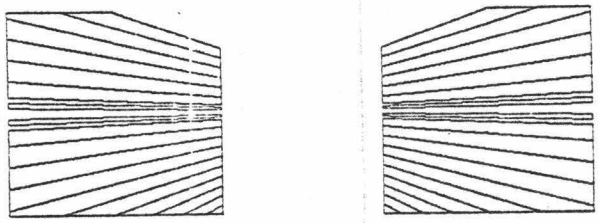
Binocular depth planes

The distance of a depth plane from the images' plane is inversely proportional to the depth plane's disparity.

If the stereo images are not coplaner, the cameras are converging on a point short of pseudo-infinity. If the photoreceptors are non-uniformly distributed as they are in the human retina, visual acuity can be enhanced by "camera convergence motors" which drive fusion within narrow width ranges in the fovea. When fusion confirms convergence, the cameras' positions can be used to determine the depth for disparity 0.

For cameras with uniform receptor densities, the only advantage of a convergence capability would be to increase the "visual coverage" of very close visual domains. Alternatively, coplaner cameras that slide closer to each other under motorized control would have the same effect without complicating the stereo geometry.

When cameras rotate to converge at a finite distance, there is *vertical* displacement to disparity in addition to the horizontal. The planes defined by the "feather of triangles" intersect the images at acute angles. Computing the intersection of rays is not difficult. The important difference is that a feature in one image may have a height different than that of its stereo counterpart. Feature match evaluation becomes more difficult when there are more possible perspective differences to features.



Converging cameras' images of a stack of planes

III C. PERSPECTIVE DIFFERENCES

The distance between the viewer and the visible objects is usually greater than the distance between the eyes. If this were not the case, small objects would be viewed from two very different perspectives, limiting fusion to only a small area of the object's image. Our eyes' *angle of convergence*, which is the same as the angle between converging stereo rays with coplaner cameras, at the close reading distance of 30 cm (about 1 foot) is only 11 degrees.

The effects of the difference in perspective between stereo views can be categorized as follows:

- Geometric variance: shape and position
- Differences in reflected light observed
- Occlusion

The following discussion of these effects forms a basis for the criteria used during Feature Match Evaluation, the next section.

GEOMETRIC VARIANCE: SHAPE AND POSITION

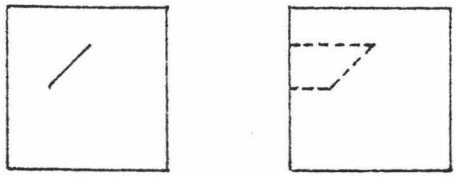
The only difference in form between the stereo images of a reflecting surface may be a difference in the images' width. Image height is invariant. Height invariance is a direct consequence of the "feather of triangles" model of the binocular space, page 39. Image width varies between views when the surface's horizontal boundaries are not parallel to the plane containing the stereo images.

When a surface is visible to both cameras, its position in one image must be within the *range of convergence* of its position in the other image. Specifically, if the west-most column of a feature in the

right image is n, then the west-most column of the feature's stereo counterpart in the left image must be *n* or greater.

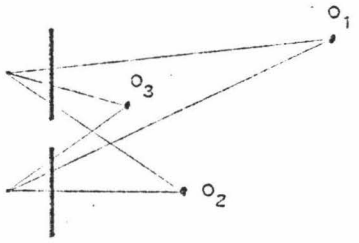
> To minimize confusion, "east"/"west" refers to the left/right sides of features within one of the images. "Left"/"right" is reserved for discriminating between cameras.

Given the feature in the left image below, its stereo counterpart may be present only in the dotted region in the right image.



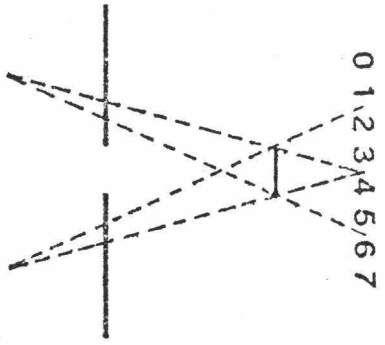
Geometric Constraints

A consequence of the horizontal shift of features is that horizontal disorderings of features are possible. In the left image of the following figure, O_1 is to the west of O_2 and O_3, but in the right image O_1 is between O_2 and O_3.



West/east feature disorders

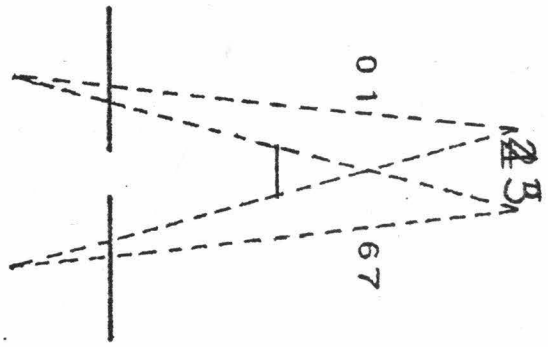
The human vision system seems to be incapable of simultaneously fusing disordered features. If fusion of disordered features were possible, illusionary surfaces would be created by the fusion of occluded features. To illustrate this point, consider the following image of numbered features and a sheet. The sheet occludes different numbered features in each image.



Cameras and scene

Left Right

Correct matches are 0_l- 0_r, 1_l- 1_r, 6_l- 6_r, 7_l- 7_r, and $SHEET_l$- $SHEET_r$. The features unmatched because of occlusion are 2_l, 3_l, 4_r, and 5_r. Matching 2_l and 3_l to 4_r and 5_r, because of the lack of competition, would ruin the west to east ordering of the fused features, defining an illusionary surface behind the line of numbered features.



Apparent scene, allowing disorder

DIFFERENCES IN REFLECTED LIGHT OBSERVED

An opaque surface is illuminated directly by light emitters and indirectly by reflections from other surfaces. To see an opaque surface is to intercept light reflected from that surface.

For models relating surface shading and camera images, see Newman and Sproull[11] or Tuong-Phong[16] who synthesize graphic images given mathematical descriptions of scenes; or Horn[5] who begins with a monocular image and then reconstructs the scene based on apparent shading.

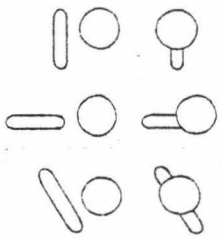
There are two components to reflected light: diffuse and specular. In general, rough dielectric surfaces reflect diffusely and smooth metallic surfaces reflect specularly. For stereopsis, the important distinction between diffusion and specularity is that only the specular component of light depends on the *angle of view*—that is, the angle between the surface normal and the eye-to-surface ray. Consequently, stereo perception of specularly reflected light will produce an illusion of depth different from that of the reflecting surface.

In conclusion, features such as shadows, shade, edges, etc. produced by a surface that reflects diffusely provide accurate stereo information because their intensities in the image and projected positions on the surface are independent of the viewing angle. However, a surface which reflects somewhat specularly may produce slightly varied features between stereo views. A surface whose reflected light is predominately specular, like a mirror, may produce stereo information which is hopelessly incorrect.

OCCLUSION

A surface visible to one image may be partially or completely occluded in the other because of the varying horizontal shift of features in stereo images. The occluders are other surfaces in the visual space or the inner edges of the camera frame: the left image's eastern boundary and the right image's western boundary.

When so-called "geometric invariants" of binocularity are violated, the only excuse is occlusion. Three types of violations to a matched pair of features are unambiguous:



Vertical extent of features unequal or not aligned

Diverging (i.e., outside of range of convergence)

Fragmentation

Whenever a match violates a geometric invariant, its excuse depends on the depth of the neighboring feature(s) that occlude. Excuse satisfaction is simply that the occluder has been matched at a closer depth than the occluded.

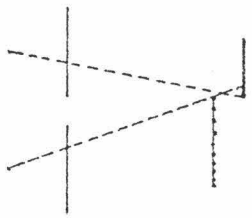
If a feature is visible in one image and *completely* occluded in the other, the lone feature's depth is unknown.

If a feature is visible in one image and *partially* occluded in the other so that

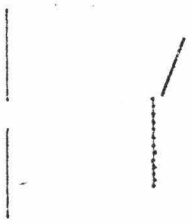
1) the occluder defines part of the occluded's border and

2) their match does not violate a geometric invariant,

then the partial occlusion is not detectable.

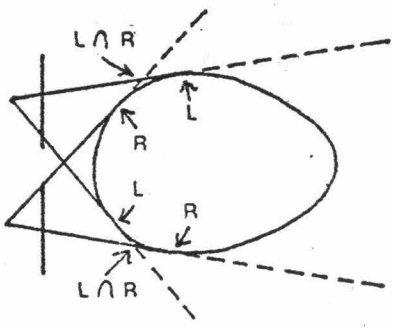


Partial occlusion of bar by dotted surface



Apparent form

The occluder and partially occluded may be the same feature: an image of a smoothly curved surface that *self-occludes*. Consider an egg. The shade is smooth over all of the surface except for the front area normal to the line of view. This donut shaped area of shade may be a single feature in a low resolution image. Since the left image's view of the egg's west side is occluded from the right image by the surface of the egg, the eye-to-surface rays will not intersect on the egg's surface.



Self-occluding curved surfaces

This is an exaggerated example because the object is sharply curved and nose-close, but self-occlusion may create subtle illusions because it is a type of partial occlusion that is not generally detectable.

III D. FEATURE MATCH EVALUATION

The fusion process decides to make or break a match based on the quality of that match, the qualities of competing matches, and the qualities of neighboring matches that will be affected by the change. The qualities of matches are determined by the *match evaluator*.

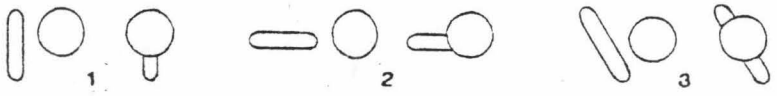
The numerical measure of a match's quality is *doubt*, so named because it primarily measures differences between stereo images and the best difference is 0: minimum doubt.

Total doubt is the sum of context-free (or local) doubt and contextual doubt. Doubt is separated this way for efficiency. Local doubt, as its name suggests, needs to be computed only once during the life of a match because its value is independent of the states of its neighbors. The qualities of a good match listed below are the match evaluator's criteria for local and contextual doubt:

CONTEXT-FREE MATCH CRITERIA

Geometric Invariance

- same top and bottom lines of image planes, (except 1)
- within acceptable width range on same line, (except 2)
- closure preserved, (except 3)



Tie Breakers

- similar average intensity
- similar internal intensity change
- disparity changes smoothly

CONTEXTUAL MATCH CRITERIA

- smoothly cohesive with neighbors
- occlusion excuses satisfied, (types 1, 2, 3)

"Is the match possible?" is the only yes/no test used by the match evaluator. The test for possibility is simply geometric: at least one pixel from the match's feature(s) in the left image must be fusable with at least one pixel from the match's feature(s) in the right image. Matching dark features to bright features, short ones to tall ones, etc. are doubtful—but may still be possible.

The range of total doubt is [0:20], where 20 would be an extremely doubtful match. Local doubt and contextual doubt are equally weighted, both having a range of [0:10]. The upper bound on total doubt, 20, is obviously arbitrary. Once chosen, however, the computational models of the match criteria can be numerically related.

Although features are matched to features—that is, areas in one image are matched to areas in the other image—they are fusable on a line by line basis only. This follows directly from the geometry of binocularity. As a result, the match evaluation procedure compares the features in a match on a line by line basis.

The doubt of a match is actually the doubt of the match's average line in the stereo images. The match evaluator takes the doubt summed over the lines in the match and divides it by the height of the match. Thus, each line within the vertical extent of the match's stereo images has a capacity for local doubt of $10/H$ and a capacity for contextual doubt also of $10/H$. The height of a match, H, is defined:

$$H := (\text{BOTTOM}_L \max \text{BOTTOM}_R) - (\text{TOP}_L \min \text{TOP}_R)$$

where the image line count is top-down and where TOP_i and BOTTOM_i are the top-most and bottom-most line numbers of the match's features in image i.

Matching taller features produces more stereo information and establishes larger stereo contexts than matching shorter features. Further, larger features—particularly taller ones—are generally more distinctive, thus less ambiguous and easier to match. However, the taller a feature, the greater capacity its match would have for doubt if doubt were not normalized for height.

For simplicity, the following description of the doubt calculation will ignore features with multiple vertical boundaries. I assume there may be features with the shape and orientation of an "E" (without serifs), for example, but "M" shapes are not included. Evaluating the match of an "E" shaped feature is basically the same as evaluating the sub-match of one horizontal leg of a "M" shaped feature. Appendix 3 contains an analysis of multiple vertical boundaries and an outline of the computation that evaluates their matches.

LOCAL DOUBT

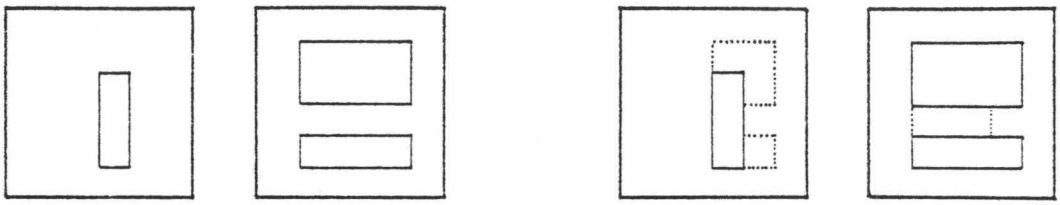
There are two parts to the criteria of local doubt: geometric invariance and tie breakers. The so-called invariant criteria have exceptions for occlusion as noted by case. Since the criteria are context-free, the "geometrically perfect" match will be preferred over the partially occluded match—assuming all other doubts are equal—even when occlusion excuses are available. Tie breakers are heuristics for discriminating between competing matches.

The measure of the geometric invariance part of local doubt is a measure of the amount of the match's *virtual* area in the stereo images which "must be occluded". The virtual images are always larger than or equal to the actual images of a match. The heights of the left and right virtual images are both *H* which, by definition, may be taller than the actual images. Running down the geometric invariance criteria, a match's virtual image may differ from its actual image in three ways:

1. The top line number of the virtual left and right images is (TOP_L min TOP_R). The bottom line number of the virtual left and right images is ($BOTTOM_L$ max $BOTTOM_R$).

2. When the eastern/western edges of the matched features diverge, the virtual eastern/western edge of the "must be occluded" feature is located to the east/west of the actual edge, widening the feature so that the edges converge.

3. When a group of disconnected features is matched to one or a group of connected features, the disconnected parts are virtually spliced together.



Sample 1 to 2 feature match

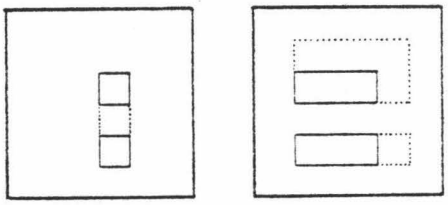
Virtual additions

The range of the geometric invariance part of local doubt is [0:10], where 10 would be the geometric doubt of an impossible match. Each line of the match that "must be occluded"—that is, each line that is exclusively virtual in one image—has doubt $10/H$. When only part of a line must be occluded, then it is proportionately doubtful:

$$\text{doubt} := \text{doubt} + (10/H) * (W_O/W_V)$$

where W_O is the number of the line's pixels that must be occluded and W_V is the number of pixels in the virtual line. Each line of each image has its own W_O and W_V. When W_O is zero, there is no doubt because there are no exclusively virtual pixels. Note that although the potential doubt for each line of each image is $10/H$, the upper bound on the total is 10 and not 20.

The tie breakers component of local doubt applies to the fusable area in the actual images, that part which is geometrically doubtless. The fusable parts of the stereo pair above are enclosed by solid lines in the next illustration.



Fusable parts

The criteria "similar average intensity" and "similar internal intensity change" specify that stereo pairs of features with same light intensities and distributions within their images make good matches. The best way to compute the doubt of a match's intensity differences is to compare the fusable parts on a line by line basis. Exact equality of intensities—particularly on a one pixel to one pixel basis—cannot be expected for two reasons: specular reflection and sub-resolution uncertainty. The latter is a subtle effect of the digital image's discreteness. When the shift of the scene's image between stereo views is not an integer multiple of a pixel width, a pixel's intensity in one image may be redistributed among two or more pixels in the other image. For example, an edge in the scene's image may fall between two pixels in the left digital image but intersect a single pixel in the right digital image. The final argument against pixel to pixel intensity comparisons is the fact that a stereo pair of features may differ in width. Only average intensities and horizontal, west-to-east trends of intensity change within fusable lines should be correlated.

To keep the use of storage low in the implementation, I simplified the doubt computation for intensity differences by comparing stereo pairs of features area to area. Local doubt, applying intensity criteria, is simply a comparison of features' average intensities and

shade vectors:

doubt := doubt + *P* * (SHADE_DIF + INT_DIF * (1 - ARROWS_PER_PIXEL/4))

The expression to the right of "*P* *", above, has a possible range of 0 to 10. The factor *P* in the doubt calculation scales the upper bound, 10, down so that only the fusable space is evaluated. *P* is the percentage of the virtual images' area that is fusable, computed on a line by line basis, each line with equal weight 10/*N*. Thus, if the doubt of the match based on the geometric invariance criteria is 2.5, then *P* would be 7.5/10.

SHADE_DIF, the difference in shade vectors, and INT_DIF, the difference in average intensities, are the two components of the intensity criteria. Their separate computations are well defined below. Their interaction, however, as I implemented it, is difficult to defend. Certainly, their relative importance depends on the amount of intensity change within the parts of the match. If the features of the match are regions of uniform intensity, with no arrows and thus no shading, then INT_DIF ranges from 0 to 10. If there are arrows, intensity difference is not the exclusive criterion. I simply chose to scale INT_DIF linearly from the "no arrows" case to the "full of arrows" case and, in this latter case, I chose to give SHADE_DIF and INT_DIF equal doubt ranges of 0 to 5. "(1 - ARROWS_PER_PIXEL/4)" is the scaling factor. As explained below, ARROWS_PER_PIXEL ranges from 0 to 2. So the scaling factor ranges from .5 to 1—the fewer arrows per unit area, the greater the factor.

SHADE_DIF is the sum, for the four arrow directions $i := 1 .. 4$, of the following:

$$|\#_{L,i} * I_{L,i} - \#_{R,i} * I_{R,i}| * 10 / (4 * TOTAL\_AREA * 255)$$

where $\#_{L,i}$ is the count of arrows in the left image that point in direction i and where $I_{L,i}$ is the average intensity of the arrows. 255, that is 2^8-1, is the maximum possible intensity difference for 8 bit pixels. When the match consists of multiple features from an image and they are touching neighbors, arrows are conjured up along their interface and added to the composite shade vector used in this computation. The orientations of the boundaries and intensities of the neighbors determine the directions and intensities of the arrows.

Where INT_L and INT_R are the average intensities of the match's features in the left and right images, INT_DIF is simply:

$$INT\_DIF := |INT_L - INT_R| * 10 / 255$$

ARROWS_PER_PIXEL is the average number of arrows per pixel within the match's features, including both the left and right images. Specifically, it is the sum, for $i := 1 .. 4$, of the following:

$$(\#_{L,i} + \#_{R,i}) / TOTAL\_AREA$$

Each pixel can have up to 4 gradient arrows since there are four sides to each square pixel. However, since each arrow internal to the feature is shared between two neighboring pixels, the upper bound on the value of ARROWS PER PIXEL is 2.

The disparity smoothness criterion for the local doubt computation is discussed later in this section with the intermatch disparity smoothness criterion. See page 61. Roughly speaking, the disparities along the vertical boundaries are smooth when the relative shape changes between stereo images are continuous.

CONTEXTUAL DOUBT

The *context* of a match consists of the neighboring features in the stereo pair of images that are nearest the borders of the match's features. The scope of such contexts varies in shape, size of area, and number of features. As implemented and defined in II E. Spatial Network, the neighbors in a context are the *outsiders* and *insiders* of the feature in the match.

When the match evaluator measures contextual doubt, it measures the *fit* of a match in its context. The dual criteria for a good fit in a context are:

- The match and its neighbors' matches define a cohesive surface across which the disparity is smooth.
- If the match being evaluated violates a geometric invariant, then some neighbor that could be occluding is in fact matched at a closer depth.

A 3-D jigsaw puzzle analogy is useful when thinking of the contextual criteria. Cohesion and occlusion excuses depend on the neighbors being currently matched. The matched parts of a context define a set of "current surfaces" in the visual 3-Space. When a context is cohesive, the surfaces defined interlock like the pieces of a jigsaw puzzle. The test for interlocking surfaces is simply that the western/eastern neighbors of the match's features in the left image are matched to the western/eastern neighbors of the match's features in the right image, on a line by line basis. Note that this condition is independent of the actual disparities and, consequently, of the depths of the matches in the context.

When neighbors satisfy the cohesion or occlusion tests for a match, the doubt of their being a good fit for the match is determined from the local doubts of their matches. For the following description of the measure, let *L* be the local doubt of the match of the neighbor in question.

To measure cohesion:

The neighbor on each side of each line of the match's features in the left image is compared to the neighbor on the same side of the same line in the right image. If the two neighbors are bound to the same match, the doubt of that 1/(2*H) part of the context is *L*/(2*H). If the neighbor pair is not cohesive with the match being evaluated—that is, they are bound to different or no matches—then the doubt is the maximum: 10/(2*H). Side(s) of a line which are unfusable are not measured for cohesion. Rather, the measure of an occluded side's fit in the context is the credibility of its occlusion excuse.

To measure occlusion excuses:

The neighbors that qualify as occlusion excuses are those which are in the match's exclusively virtual image. An excuse holds for a side of a line if the neighbor is matched with a disparity that is greater than or equal to the "virtual disparity" of the occluded line's side. If only one side of a line must be occluded, then there is only one possible excuse—only one neighbor. But if an entire line must be occluded, there may be multiple excuses—multiple neighbors in the exclusively virtual line along the horizontal borders of the match. As implemented, only one satisfying excuse for a line is needed. Where *L* is the minimum local doubt among the line's excuses which hold, the doubt of the occlusion excuse is *L*/*H*. When no excuses hold, the doubt is the maximum: 10/*H*. When only one side of a line must be occluded,

the doubt is $L/(2*H)$ if the excuse holds and $10/(2*H)$ otherwise.

As implemented, the "virtual disparity" used to compare with the disparities of possibly occluding neighbors is the minimum disparity of the occluded match. A more discriminating virtual disparity for large features could be derived by using the nearby fusable boundaries of the match to project a linear trend of disparities across the exclusively virtual line.

Either neighbors are cohesive or they are not; either a neighbor is possibly occluding or it is not. Since the decision is made at the "atomic level"—line by line and side by side—two multi-pixel neighbors may be *partially* cohesive or a neighbor may be partially occluding another, but there is no concept of *nearly* cohesive or nearly occluding. All non-cohesive parts of contexts and unsatisfied occlusion excuses are equally doubtful.

The measure of contextual doubt is *skeptical* because the unknown context is as bad as the worst context. At any one time during the fusion process, none or only some of the neighbors in a context may be matched. When a neighbor is not part of a match, the match evaluator assumes that the neighbor's stereo counterpart is occluded, making the neighbor unmatchable. Unmatched features break the continuity of the context's disparity and, of course, are not an occlusion excuse. The consequences of skepticism in evaluation are that 1) making a match cannot increase the doubts of matches in its context and 2) breaking a match cannot decrease the doubts of matches in its context.

DISPARITY SMOOTHNESS.

Smoothness in a match's disparity is a local criterion for match quality and smoothness in a cohesive context's disparity is a contextual criterion for match quality. The basis of these criteria is the assumption that most interfaces between features are lines in areas of continuous disparity change.

The primary discontinuity in intermatch disparity is a jump in depth between neighboring surfaces. An edge between neighboring surfaces or a rumple in one surface's depth continuity will usually be a feature in itself, fragmenting the image of the surface(s) into two or more features. This follows from the definition of features and physical models of surface shading. If, however, the boundary of a surface intersects a single feature in one image and the surface's boundary is distinctive in depth, then splitting that feature during form modeling—following fusion—defines a "subjective contour". Of course, a true subjective contour would be defined by splitting the feature following 2-D recognition.

The local criterion for disparity smoothness is that disparity changes nearly linearly along the vertical boundaries of matched features. Thus, for example, matches of "|" to | or \ or / are equally doubtless and preferred over a match of | to > . If d_i, d_{i+1}, d_{i+2} are the disparities of the match's western boundary for lines i, $i+1$, and $i+2$, then the doubt of disparity smoothness about the center line $i+1$ is proportional to DIF_W:

$$DIF_W := |(d_i + d_{i+2})/2 - d_{i+1}|$$

Similarly, a three line segment of the match's eastern boundary produces

DIF_E. The total roughness of disparity for a line is then DIF:

$$DIF := DIF_W + DIF_E$$

To integrate this with the other local doubt computations, W_O and W_V are given subscripts for the left and right images: $W_{L,O}$, $W_{R,O}$, $W_{L,V}$, and $W_{R,V}$. "L" and "R" mean left and right; "O" means occluded; "V" means virtual. $W_{L,O}$ is the number of pixels in line $i+1$ of the left image which must be occluded according to the geometric invariance criteria and $W_{L,V}$ is the total number of pixels in virtual line $i+1$ of the left image. Introduce: $W_{L,f} = W_{L,V} - W_{L,O}$ and $W_{R,f} = W_{R,V} - W_{R,O}$. $W_{L,f}$ and $W_{R,f}$ are the numbers of pixels in the virtual lines which are fusable. Then, the local doubt of disparity smoothness for line $i+1$ is:

$$doubt := doubt + (10/H)*((DIF/2 \min W_{L,f})/W_{L,V} + (DIF/2 \min W_{R,f})/W_{R,V})$$

Disparity roughness cannot be attributed to either the left or right image alone so both are discredited by an equal amount: DIF/2.

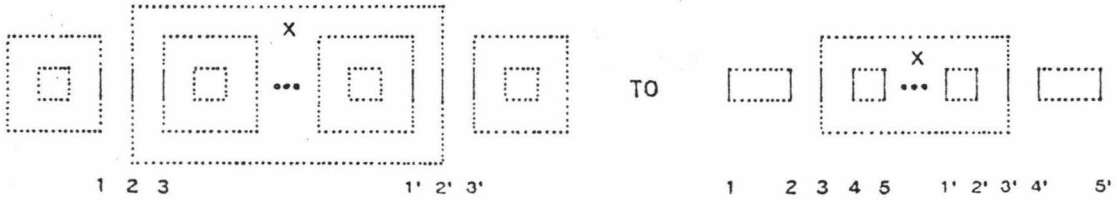
> After this part of local doubt has been computed, the P in the computation of local doubt that applies the intensity criteria must be adjusted to reflect the remainder of a line's potential doubt that is available. See page 55.

Besides inhibiting random matches, the effects of the local criterion of disparity smoothness are to encourage patching up holes in the sides of fragmented features and to prefer "syntactic" contours over "subjective" contours.

Similarly to the local criterion, the *contextual criterion* for intermatch disparity smoothness is that disparity should change nearly linearly. However, instead of measuring disparity smoothness along the vertical boundaries of the match, it is measured *across* the vertical

boundaries to cohesive neighbors. The test is made on a line by line basis, each line providing either one trend of 4 to 6 disparities or two trends of 3 to 5 disparities, depending on the presence of *insiders*. The following illustration of different context configurations for a match's line explains what possible trends are available for the smoothness test. X marks the match's region.

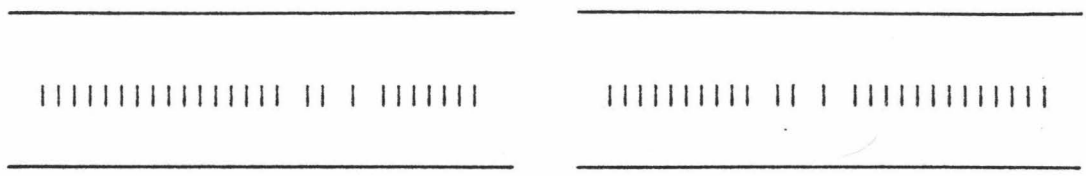
One trend of 4 to 6 disparities



Two trends of 3 to 5 disparities

The disparity trends may be shorter if neighbors are not cohesive with X because they do not participate in the smoothness test. The lengths of trends in completely cohesive lines varies depending on the presence of *insiders*. If a cohesive neighbor—*outsider* or *insider*—has an *insider*, then only the cohesive neighbor's nearest border participates in the smoothness test.

Although the cohesion test alone is powerful, a "random bar stereogram" can be correctly fused only by using the smoothness test:



The cohesion test for a match is satisfied when all of the match's neighbors are matched and matched in the same west-to-east order in both images. If cohesion were the only contextual criterion, then there would be many equally good fusions of the images above.

I did not implement a test for intermatch smoothness. There are two reasons: 1) the cohesive criterion alone is almost sufficient and 2) the spatial network representation of the image inadequately relates neighboring *insiders*. See page 36.

THE MATCH'S DATA STRUCTURE: *STEREGION*

When the match evaluator is activated, it is passed two feature lists and an upper bound for acceptable doubt. The lists, one for each image, point to the features in the hypothetical match to be evaluated. The upper bound for doubt is used to terminate the evaluation of a losing match early. During the evaluation process, doubt accumulates. If the match is possible and the final doubt is less than the upper bound, then the match evaluator returns to the caller the following information about the match:

- Local and total doubt
- Minimum disparity among the fusable boundaries
- List of neighbor pairs for the cohesion test
- List of occlusion excuses

If the caller of the match evaluator decides to make the match in question, then the match is established in the system's global 3-D network. The match is characterized by a record data structure called *STEREGION* which connects the two spatial networks of the two stereo images via pointers to the parts of the match. Concurrently, each feature's data structure, REGION, is modified to point to the STEREGION—thereby binding the features to the match.

STEREGION:

<LEFT_PARTS>, <RIGHT_PARTS>
LOCAL_DOUBT, TOTAL_DOUBT
MINIMUM_DISPARITY
<NEIGHBOR_PAIRS>
<OCCLUSION_EXCUSES>
NEEDS

<LEFT_PARTS> and *<RIGHT_PARTS>* are lists of pointers to the REGIONs in the left and right images that comprise the match. The minimum, of course, is one from each image. Conversely, the <STEREGION> subfields of the specified REGIONs point to the STEREGION.

LOCAL_DOUBT and *TOTAL_DOUBT* are real numbers. TOTAL_DOUBT is a variable because it changes with the context.

MINIMUM_DISPARITY, an integer, is used for evaluating occlusion excuses—both its own and its neighbors' excuses.

<NEIGHBOR_PAIRS> is a list of the neighbors in the match's context, paired off for the cohesion test. The list's format is:

DOUBT_FACTOR
<L_REG>, <R_REG>
<NEIGHBOR_PAIRS>

<L_REG> and <R_REG> point to two neighboring REGIONs, one from each image. When the match is cohesive with this part of the context, L_REG is matched to R_REG. If the local doubt of their match is *L*, then their part in the contextual doubt of the match in question is *L* * DOUBT_FACTOR. If L_REG is not matched to R_REG, then the doubt is 10 *

DOUBT_FACTOR. DOUBT_FACTOR is the proportion of the match's vertical boundaries across which L_REG is a neighbor in the left image and R_REG is a neighbor in the right image, in correspondence on a line by line basis. (Info: page 59).

<OCCLUSION_EXCUSES> is a list of excuses needed for the match's geometric imperfections. Its format is:

DOUBT_FACTOR
<OCCLUDING_REGION_LIST>
<OCCLUSION_EXCUSES>

Each of the <OCCLUSION_EXCUSES> is evaluated independently, each weighted by a DOUBT_FACTOR. DOUBT_FACTOR is the proportion of the match's virtual height which must be occluded by the REGIONs in <OCCLUDING_REGION_LIST>. (Info: page 59). Among the regions in the list that are currently matched with a minimum disparity greater than or equal to the occluded match's MINIMUM_DISPARITY, the least doubtful locally is chosen as the best excuse. If this neighbor's local doubt is *L*, then the contextual doubt of the occlusion excuse is *L* * DOUBT_FACTOR. If none of the REGIONs in <OCCLUDING_REGION_LIST> qualify as an excuse, the doubt is 10 * DOUBT_FACTOR.

NEEDS is an integer variable used during the fusion process. See the next section of this chapter.

III E. FUSION PROCESS CONTROL STRUCTURE

The objective of the fusion process is to fuse the stereo images —to match all fusable features correctly. I think the algorithm I present here favorably compares with human vision systems in meeting this objective on a stereo snapshot basis.

To experience stereopsis for understanding, the reader should consult Julesz's *Foundations of Cyclopean Perception*. Julesz has pioneered human stereopsis, establishing its major investigating tool— the random-dot stereogram.

I describe the fusion process below in five parts: Discussion, Global Data Structures, Process Overview, The Details, and It Halts.

DISCUSSION

The fusion process I present here is sequential, but its task of constructing the apparent scene is performed by *concurrently* fusing independent parts of the stereo images. That is, independent groups of features in the images will usually be fused during the same time interval. Fusion progress would be illustrated best by an animation of the apparent scene as surfaces appear and disappear, as matches are made and broken—one change per movie frame. Seeds for stereo contexts, matched features with unmatched neighbors, may appear at any time as lone surfaces. When separate groups of contiguous—actually intersecting—stereo contexts grow to eventually "touch", the newly neighboring matches interact.

Construction of the apparent scene, fusion of the stereo images, progresses along general trends:

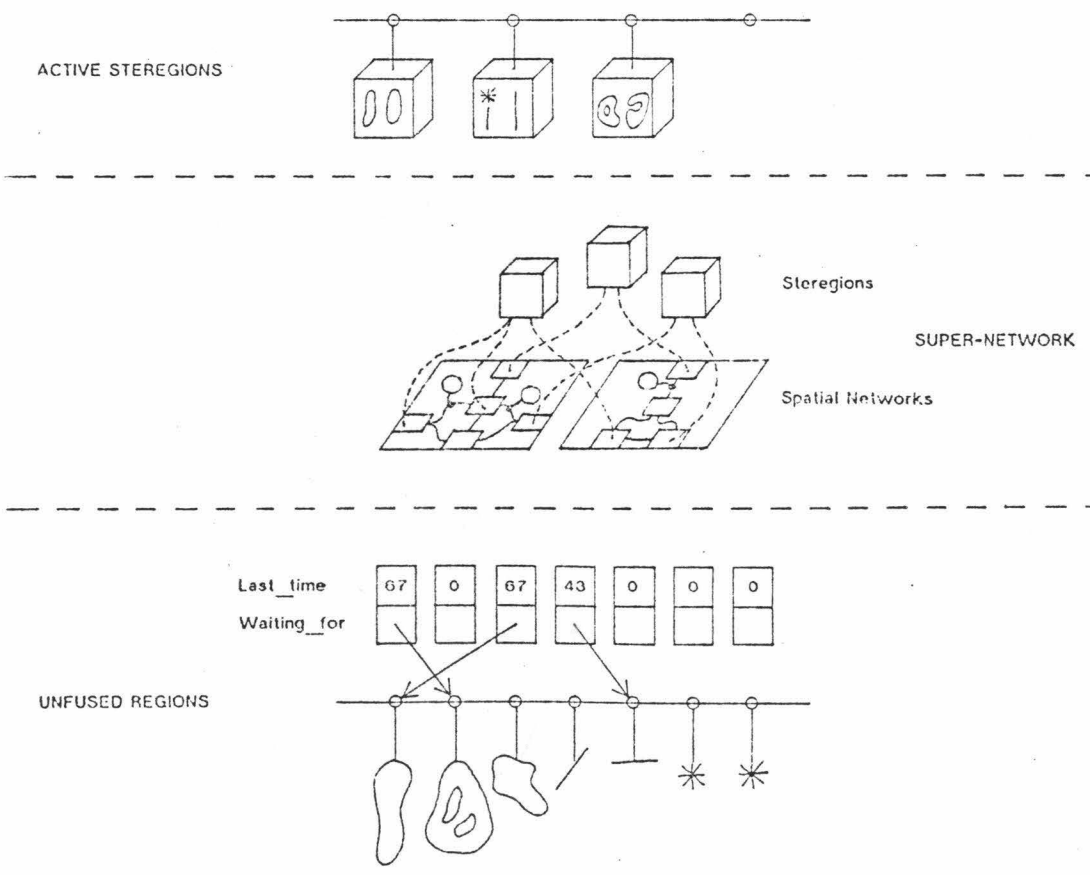
- Unique to ambiguous
- Defined stereo contexts to undefined stereo contexts
- Tall to short
- Large to small
- Front to back, where rear features are partially occluded

These are the trends for the process I describe here. Note, however, that this process always begins from a "cold start". By guiding the fusion process with anticipations of the scene, the trend of fusion—*known to unknown*—would provide "hot start" efficiency. Identification of objects in the scene and extensive memorization of forms are unnecessary for anticipation. A stereo movie process could simply use the apparent scene computed from the last stereo pair of frames to fuse the current pair.

GLOBAL DATA STRUCTURES

EYES, the fusion process, begins when it receives a stereo pair of spatial networks of regions from L_EYE and R_EYE. EYES fuses the images by building a super-network of steregions that connects the two spatial networks. Throughout the life of the process the super-network represents the current state of the apparent scene's development.

In addition to the super-network, EYES uses two lists: *active* steregions and *unfused* regions.



"Active steregions" is a list of matches. They are ordered by total doubt, least first. The matches on the list wait to be considered for growth, for expansion to include a neighboring feature that improves the match. A 1-to-1 match—one feature in the left image matched to one feature in the right image—for example, may become a 1-to-2 match.

"Unfused regions" is a list of all the unmatched regions in the left and right networks. Initially the list points to all regions. When the process terminates, the list will usually point to few or none. Associated with each region pointer in the list are two items of information: *last_time* and *waiting_for*. Last_time is an integer. The process uses this time variable to remember "the time when I last attempted to match this region". Waiting_for is a region pointer. When assigned a value, it points to another region in the unfused list.

The order of regions in the unfused list partially determines the order in which features are matched. The regions in the unfused list are ordered first by height (tallest first), then by area (largest first), and then by location in the image (top-down, left-right). Thus if region R_A is taller than region R_B, then R_A will be closer to the front of the unfused list than R_B. If their heights are equal, then area is the discriminator. Last of all, if their areas are also equal, then they are ordered by location. This last ordering criterion will significantly affect the order only among the very short, skinny regions which will be at the end of the unfused list.

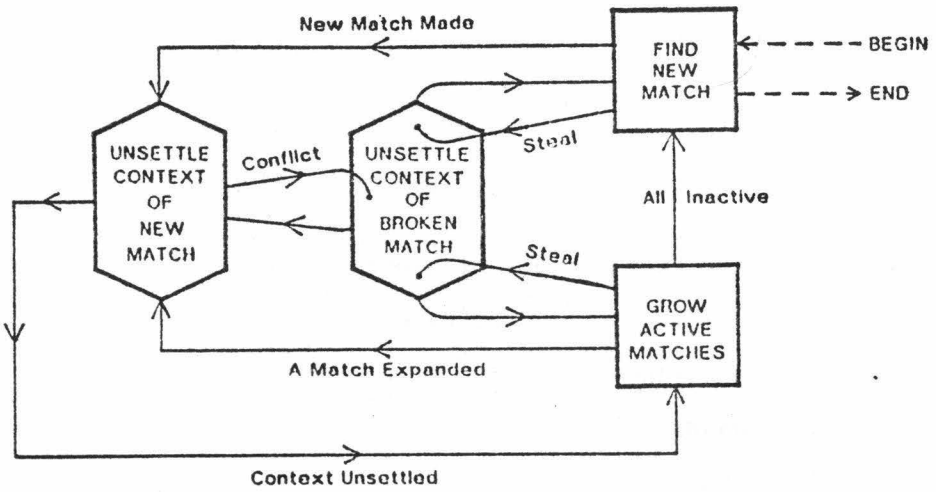
The height and area ordering criteria together approximate the single criterion: "number of vertical boundary units". Since fusion only of the vertical boundaries of regions provides depth information, regions with the most vertical boundaries have potential for generating

the most stereo information. These regions should be placed at the front of the unfused list. A region's vertical boundary size could be computed exactly by tracing the region's chain encoding and counting vertical moves, but height and area data are already available in the region's record data structure. Alternatively, by introducing a new subfield—vertical_size—to non-point regions, the precise ordering information could be made readily available.

The last global variable of the fusion process is a counter: *now*. *Now* is initialized to 0 and is incremented by one or more every time a match is made, broken, or expanded—every time the super-network is changed. This clock of fusion progress is important. When the process updates the touched_time subfield of a region record or a last_time in the unfused list, the value of *now* is assigned. By associating times with places in the spatial networks, the flow of process control can be modified dynamically by the order of events: 1) when a part of the apparent scene was last changed and 2) when a change to a part of the apparent scene was last attempted.

PROCESS OVERVIEW

The following is a flow chart for the fusion process:



By temporarily disregarding the subprocess "unsettle context of broken match" and the idea of breaking matches, the flow of control can be more easily described.

The fusion process begins with "find new match". If no new matches can be found using regions from the unfused list, the fusion process ends. If, however, two regions on the unfused list are matched or one region on the unfused list is added to an existing match, then it is recorded in the super-network by introducing a new steregion or by expanding an existing steregion. After the super-network has been changed, control passes to "unsettle context of new match".

Whenever a match is made, broken, or expanded, the context of the change is *unsettled*. There are two steps to unsettling a context: 1) activate neighboring steregions and 2) update the touched_time subfields

of neighboring regions with *now*. After the context has been unsettled, process control passes to "grow active matches".

Steregions on the active list are candidates for growth. Active 1-to-1 matches grow to become many-to-many matches, one feature at a time. The condition for growth is simple:

> A match can be expanded by absorbing a neighboring region if the action reduces both its local doubt and its total doubt.

The matches are considered for growth in the order of their active list positions, best first. As each match is considered, it is removed from the front of the list. If the list becomes empty, process control passes to "find new match" where control began. Otherwise, if a match is expanded, the networks are appropriately modified and control passes to "unsettle context of new match" before any further attempts to expand active matches.

"Unsettle context of broken match", the center of the flow chart, is executed whenever a match is broken. Like a subroutine, control always returns directly to the caller. This is depicted by the "conflict" and two "steal" loops. Matches may be broken by any one of three subprocesses:

Find New Match

A region that is already bound to a match would make a better match with an unfused region.

Grow Active Matches

A match is expanded by absorbing a neighboring region that is already bound to another match.

Unsettle Context of New Match

A match newly made or expanded conflicts with an existing neighboring match, breaking it. Neighboring matches conflict when they are not mutually cohesive.

The capability of breaking matches, if defined incorrectly, could introduce non-determinism to the fusion process. In "It Halts", below, I explain the importance of the destruction criterion in proving that the process terminates.

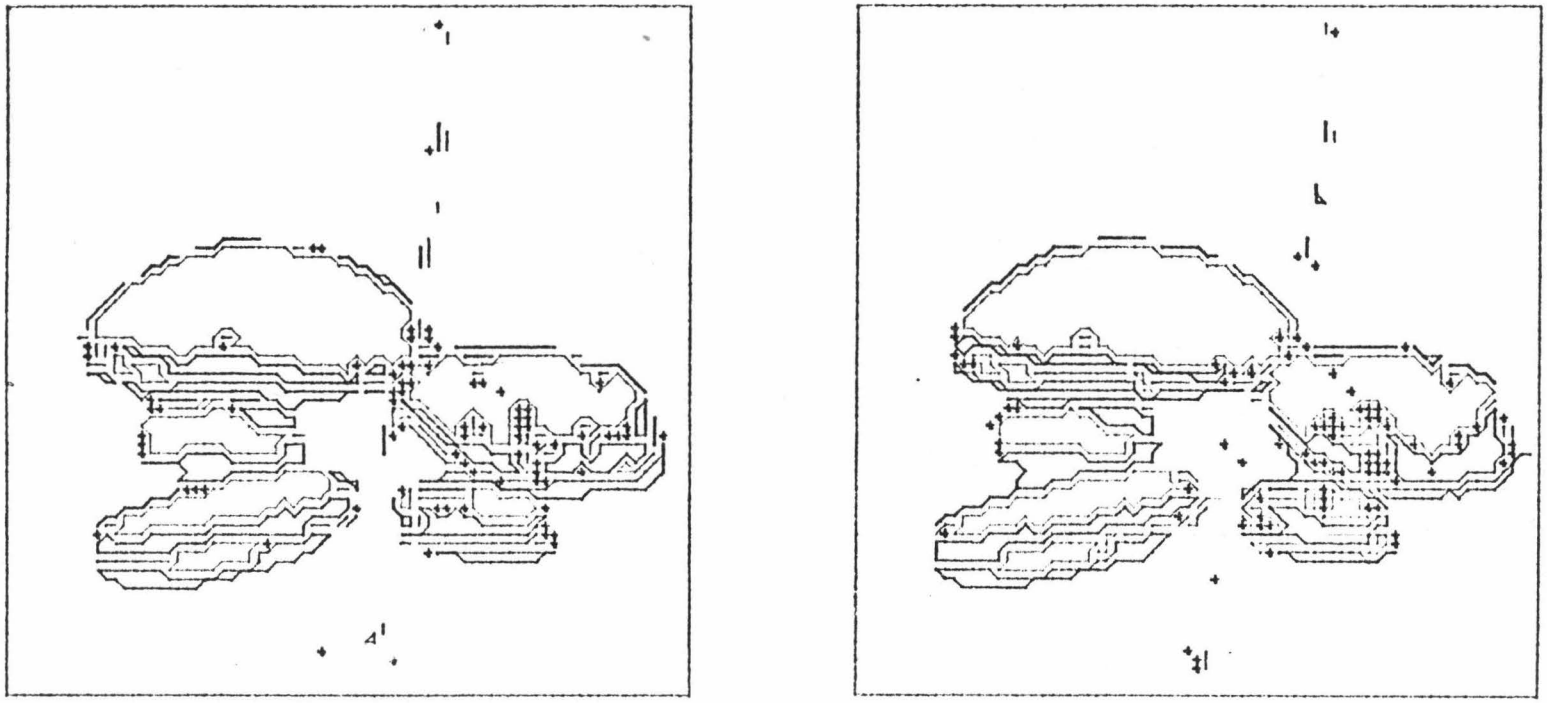
The *COMMON CONDITION FOR BREAKING A MATCH* is based on the net effect of the dual change: adding a new match and breaking another match. The criterion is simply that among all of the matches affected by the dual change, the best of the matches that improve is better than the best of the matches that worsen. More specifically, the best after-change doubt among the matches that will be improved with the change must be less than the doubt of the match to be broken and less than the before-change doubts of all matches that will be worsened with the change. Remember: a match improves when its local doubt decreases and a match worsens when its total doubt increases; the best match has least total doubt.

How do the total doubts of activated neighbors change? Making a completely new match can never increase the total doubts of neighboring matches. By expanding a match, however, the new match's local doubt may be increased—its contextual doubt decreasing by a greater amount to compensate—thus possibly weakening neighbors with additional contextual doubt. Deleting a match, the last case of change, can never improve the matches in its context when considered a separate action.

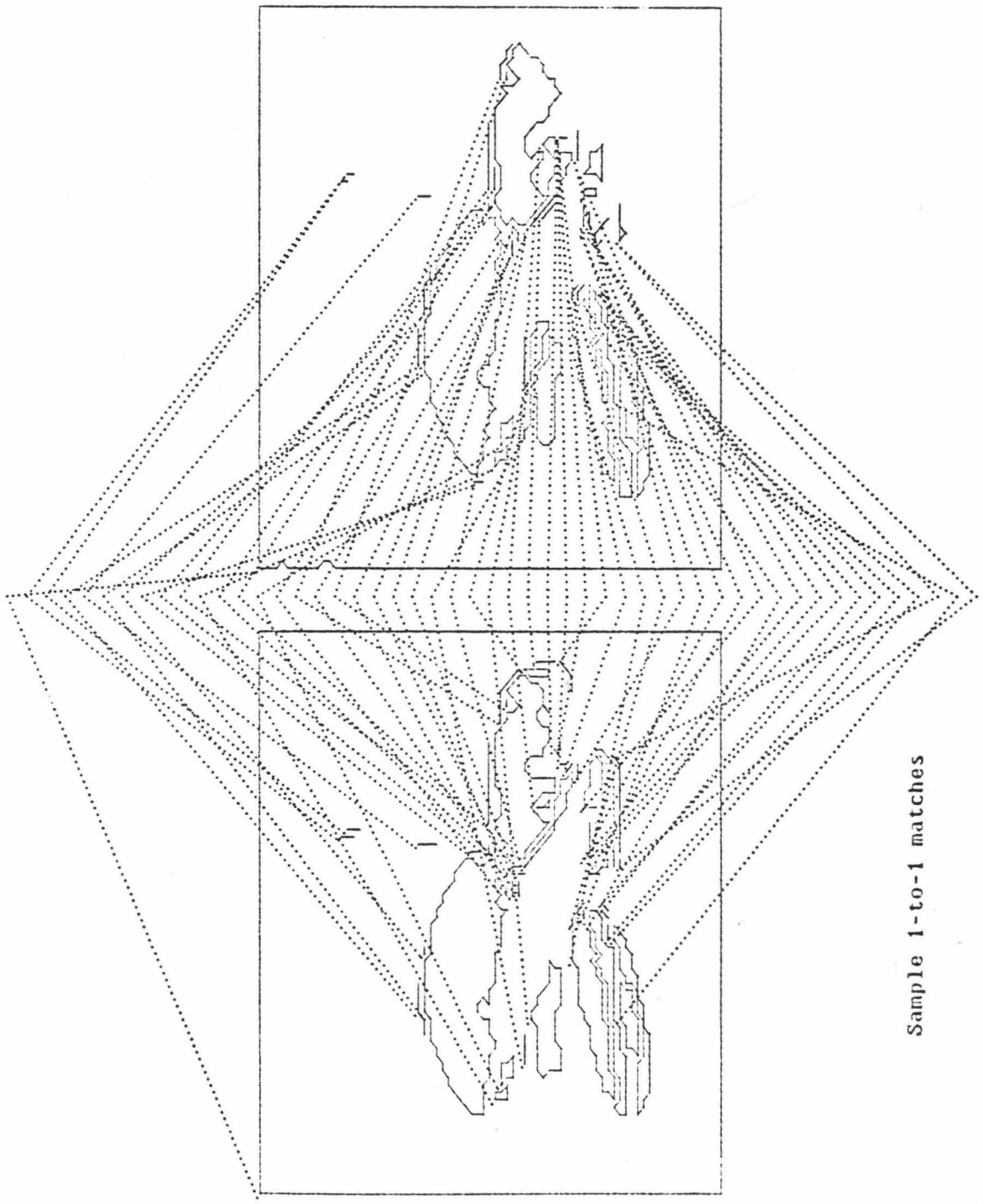
As the flow chart on page 73 illustrates, "grow active matches" and "unsettle context of new match" form a loop within the "find new match" - "unsettle context of new match" - "grow active matches" outer loop. There is an inner loop of *renovation*, or *stabilization*, within an outer loop of *innovation*. Innovation consists of taking an unfused piece of a spatial network and locating a match for it with pieces of the opposing network which may or may not already be a part of an existing match. Every modification to the super-network, such as the making of a new match, is a likely source of immediate instability to its local context. Reevaluation of this context's elements and subsequent renovation may trigger, in turn, secondary instabilities requiring new renovation, etc. Stability is achieved when each of the elements in the extended context touched by change has been reviewed in its most recent relevant state with further modification decided to be unnecessary.

I developed the stereopsis algorithm by using two debugging methods: tracing control and plotting results. Tracing control consists of sitting patiently in front of a terminal and reading messages as they are generated by the program. Plotting results consists of interrupting the program, or waiting until it terminates, and displaying the current state of fusion on a line drawing device. The next few pages contain examples. The scene consists of five partially overlapping rocks. First, the gradient arrow representations of the stereo images are illustrated. Then, the boundaries of the extracted features are illustrated. Last of all, matches are depicted by dotted lines that link features in the left image with features in the right image. Two "fusion plots" are presented, neither of which represents a complete fusion of the images. The first consists of only 1-to-1 matches. The second fusion plot includes many-to-many matches. Note that a temporal component to the plots is missing. That is, by watching the plotter slowly making the dotted links to features, I was able to recognize specific matches more clearly.

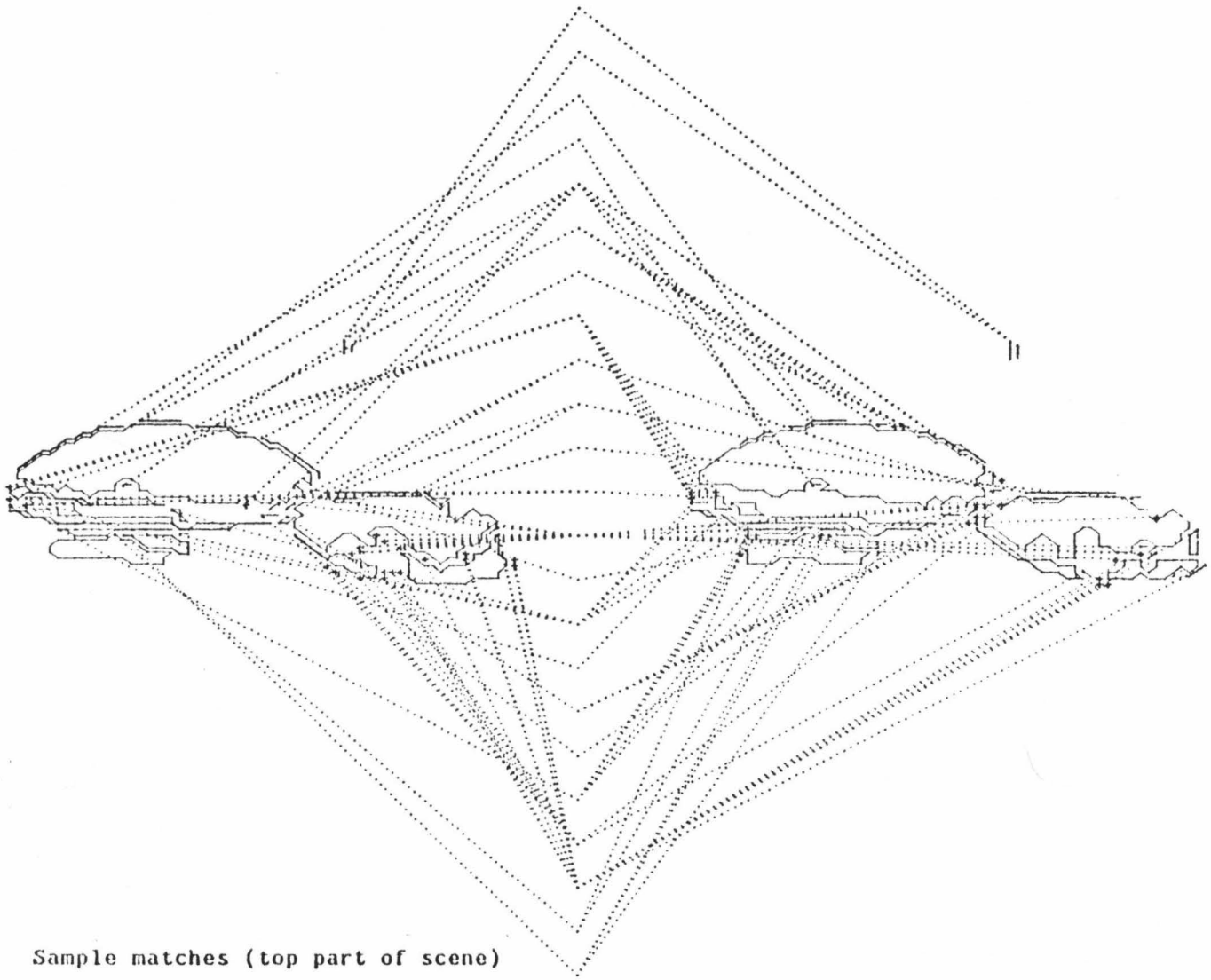
Gradient arrows



Boundaries of regions



Sample 1-to-1 matches



Sample matches (top part of scene)

THE DETAILS

Consider the fusion process to have three parts—unsettling contexts, growing active matches, and finding new matches—for the following discussion. I present refinements of the algorithm which are important for understanding its adequacy or efficiency.

Unsettling Contexts

The scope of a single change's *direct* effect is *one* "neighbor-distance", inside and out. The context of a match is defined by the neighbor relation, as introduced in II E. Spatial Network and refined in Appendix 2. When a match M_c changes—made, broken, or expanded—all of M_c's neighbors are usually affected. Just as half of M_c's doubt depends on M_c's neighbors, M_c's neighboring matches depend on M_c. When M_c is made, for instance, it will encourage unmatched neighbors in one image to be matched with certain neighbors in the other image. The new matches encouraged are those which will be cohesive with M_c or those which can use M_c as an occlusion excuse. The amount of encouragement depends on the presence and local doubt of M_c. The local doubt of M_c, in turn, depends on no other matches.

The scope of a single change's *indirect* effect is *two* "neighbor-distances", inside and out. A region R_{nn} is at a distance of two neighbors from a match M_c if R_{nn} is only a neighbor of a neighbor of M_c. When M_c changes, only its neighboring matches may change in value—specifically, their contextual doubts and, therefore, their total doubts may change. This, in turn, may affect new changes to their neighbors: the original change's neighbors' neighbors. Changes that depend on the contextual doubts of neighboring matches are those that involve breaking

matches or expanding matches. For the examples below, let M_n be a match neighboring M_c and let M_{nn} be a match neighboring M_n.

Example 1:

> M_c is broken, increasing the total doubt of its neighbor M_n. If M_{nn} had previously been prevented from stealing a region of M_n for its own, M_n's new increase in doubt may now make the theft possible. If so, M_n's match will be broken and M_{nn}'s match expanded.

Example 2:

> M_c is newly made or expanded, decreasing the total doubt of M_n. If R_{nn} is an unmatched neighbor of M_n, the new improvement of M_n may now make a match for R_{nn} possible. Specifically, if a match of R_{nn} to another region R had previously been prevented because R was already matched and the "common condition for breaking a match" was not satisfied, M_n's new improvement to R_{nn}'s context may now satisfy the condition. If so, R's match will be broken and R will be matched to R_{nn}.

When a match is changed, the context that is unsettled extends two neighbor-distances from the match's boundaries to include indirectly influenced regions and their matches. The unsettling procedure is recursive: matches are activated and time is updated through two "nested" neighbor relations.

> In fact, the unsettling procedure is "multi-recursive". In implementation, the procedures to unsettle the contexts of new and broken matches were combined. There was less code but more complexity than the system I implicitly describe here.

The depth of recursion necessary to unsettle a stereo context to a width of two neighbor-distances is managed by using the touched_time subfields of regions. With each call to the unsettling procedure, *now* is incremented. Regions are then marked as they are processed, like a Lisp garbage collecter marking Lisp nodes.

When a match is broken, unsettling its context consists of 1) activating matches and 2) updating the touched_time subfields of regions.

1. Part of the activation task is to reevaluate the contextual doubts of matches that are direct neighbors and assign new values to their steregion's total_doubt subfields. The second and last part of the activation task is to place onto the active list all "needy" matches in the stereo context. This includes both direct and indirect neighbors. "Needy" is explained below.

2. The touched_time subfields of all regions in the stereo context, including direct and indirect neighbors, are updated with *now*.

Unsettling the context of a new or expanded match is the same as unsettling the context of a broken match, with one exception: a new or expanded match can break neighboring matches with which it conflicts. All matches neighboring a new match may potentially conflict with it. A changed match M_c *conflicts* with a neighboring match M_n if M_n satisfies no occlusion excuses of M_c and there is no cohesion along the interface of M_c and M_n. The test for conflicts is simply part of the computation that evaluates contextual doubt. Once a match is found to be

conflicting with the new or expanded match, the "common condition for breaking a match" must be tested and found to be satisfied before the neighboring match is broken.

The *needs* subfield of a steregion was introduced in III D. Feature Match Evaluation, but the description of its use has been postponed until now. *Needs* is a three state variable, requiring only two bits to represent. The three states are *needs_unknown*, *need_exists*, and *needless*. They refer to the steregion's need to grow. When a match is made or expanded, *needs* is in state *needs_unknown*. After the subprocess "grow active matches" unsuccessfully attempts to expand an active match, the match's *needs* is assigned *need_exists* or *needless*. The importance of this variable for unsettling contexts is: only unsettled matches that are not in a *needless* state are actually placed on the active list.

Although a single change to the apparent scene does not immediately propagate throughout, unsettling the entire network, it may ultimately be the seed for a revolution. Changes influence changes via the unsettling procedure which activates matches and "awakens" unfused regions. A newly made match with little doubt may significantly influence a context. This context may consequently undergo higher priority refinement. Complete propagation of a revolutionary change will usually require many alternating phases of new match making and growth.

Revolutions are expensive because previously made matches are destroyed—work is wasted. Consequently, the fusion process is designed so that it usually converges directly towards a successful terminal state.

Growing Active Matches

The ability to match several features in one image to one or more features in the other image is necessary because of fragmentation. Regardless of the feature extraction technique, fragmentation will occur. The reasons are basic to stereopsis: image widths vary between stereo views and there is always sub-resolution uncertainty. Further, slightly specular reflections may cause the left and right images of a surface to be slightly different—which, in turn, may cause the surface's left and right images to divide up differently into features.

With the ability to match many-to-many, the number of possible matches explodes. Matches grow one feature at a time, however, and growth is sufficiently constrained so that stereo distinctions in the images are seldom lost by being absorbed. Otherwise, a possible consequence of unleashed growth would be a single match of everything in the left image with everything in the right image. This fusion would produce no stereo information.

The matches on the active list are processed by "grow active matches" in the following way. Starting with the best, the process tries to decrease the match's local and total doubt by expanding it with an exterior, non-enclosing neighbor. Merging insiders with outsiders is not allowed. If more than one neighbor satisfies the growth condition, the process chooses the one which most improves total doubt. The match will be considered for additional growth at a later time. First the context of the expanded match must be unsettled which may place neighboring matches closer to the front of the active list than the newly expanded match.

In order for a match to be expanded by stealing a neighboring region that is already matched, the "common condition for breaking matches"—that is, the *breakable condition*—must be satisfied in addition to the growth condition. The growth condition is that the local and total doubts of the expanded match decrease. The breakable condition assures that the sacrifice results in progress.

For efficiency, I introduce the steregion's subfield *needs*. When a match is newly made or expanded, its needs subfield is assigned *needs_unknown*. When the match is removed from the active list because no expansions can be made, it is assigned a different value. If all expansions considered failed to decrease the match's local doubt, then *needless* is the value assigned. Otherwise, if an expansion satisfies this condition but fails one of the other conditions, then *need_exists* is the value assigned. A match that has been classified as *needless* will fail all subsequent attempts to grow because it failed the local growth condition. This condition can not be changed by modifications to the context. Consequently, matches classified as *needless* are not placed on the active list when unsettled.

My last note on the efficient growing of matches involves upper bounds for acceptable doubt. The match evaluator may optionally be passed upper bounds for local and total doubt along with the hypothetical match to be evaluated. The match evaluator uses them to terminate the evaluation of the match early if the match's failure is found to be inevitable. The subprocess "grow active matches" sends the match evaluator various upper bounds. When calling the evaluator with a hypothetical expansion, the local doubt of the active match is always sent as an upper bound. If the active match's needs are unknown, then no upper bound to total doubt would be sent. Otherwise, if needs are

known to exist, the match's total doubt would be sent along with its local doubt. If hypothetical expansions have satisfied all conditions, yet not all neighbors have been considered for absorption, then the total doubt of the successful expansion is sent as an upper bound for the next expansion to be considered.

Finding New Matches

Unique features in the images should be matched early in the fusion process for good reasons:

- They are easiest to match correctly
- They are least likely to be broken later
- Correct matches establish correct contexts, leading to other correct matches

Random-dot stereograms are usually more difficult than the "average natural scene" to fuse because the features in the stereograms—essentially dots—all have the same size, shape, and intensity. There may be no striking features to "prime" the fusion process.

The unfused list is the search space of the *match finder*, so the order of the regions on the list is important for efficiency. The ordering I chose is based on the amount of stereo information the regions can potentially generate. This means that the unfused regions are basically ordered by height.

Is height the only discriminating quality of "unique features"? Consider a slightly pathological example: a fly sitting in the middle of a square which is part of a large checkered pattern. Assuming the pattern and its stereo images are otherwise *perfect*, the fly's stereo images will be the only unique features—yet they are at the end of the

unfused list! It is conceivable that the fly's features could be separated from the stereo patterns before the fusion process begins by analyzing the aggregate feature population. By computing distributions of region qualities, the "uniqueness" of sample regions could be a function of their deviations from the means: average height, average size, average intensity, and average intensity change, for examples. However, the calculations are costly and a fixed set of them will never be sufficient. New distinctions in the scene may always be discovered by perusal, demanding accommodation. Alternatively, ordering unfused regions by the amount of potential stereo information is well founded in domain independence.

The match finder begins with the region at the beginning of the unfused list. Among the possible matches for the region, the best is either made or postponed. There are three types of possible matches for an unfused region R_u:

1. R_u with another unfused region.

2. R_u with a region R that is already matched. To qualify as possible, R's match must be breakable—that is, the "common condition for breaking matches" must be satisfied.

3. R_u is added to an existing match, expanding it. To qualify as possible, the existing match must be breakable.

Note that the breakable condition is required in this third match type, expanding a match, but is not a condition for growth. Growth's own special condition requires that the local doubt of the expanded match improve. This never fails the breakable condition.

Unfused regions are matched to other unfused regions when the match is best for both. These best 1-to-1 matches are located by using the waiting_for subfield in the unfused_list. When the match finder finds that the best match for an unfused region R_u is with another unfused region R_{au}, it inspects R_{au}'s waiting_for subfield in the unfused list. If the waiting_for of R_{au} is pointing to R_u, then the match is made. Otherwise, the waiting_for of R_u is made to point to R_{au}, postponing the match. The match finder would then skip R_u on the unfused list in order to consider the next unfused region.

If the best match for an unfused region is not unique—that is, another possible match has equally low total doubt—matching the unfused region must be postponed. There may not be an unambiguous best 1-to-1 match in a perfect pattern. Although an image of even a perfect pattern will almost always be imperfect, parts of the pattern's image may still be ambiguous to match. Consider, though, the completely ambiguous case of a perfect checkered pattern. Many possible matches for a black square in one image may be found using black squares in the other image. By successively postponing match making because of ambiguity, the match finder will reach the end of the unfused list without having made any matches. At this time, a random choice must be made.

By including the ambiguous case, a match for an unfused region R_u may be postponed for two reasons:

1. R_u's uniquely best 1-to-1 match is with another unfused region R_{au}, but R_{au} is not waiting_for R_u. If so, either R_{au} has not been considered by the match finder yet or it is waiting_for a region other than R_u.

2. R_u's best 1-to-1 match is not unique.

For efficiency, I suggest that postponements of the second type also use waiting_for to point to one of the regions with which R_u makes a best match. I explain the practicality of this below. If used, however, an extra bit of information must be introduced in order to distinguish between the two types of waiting_for: unique and ambiguous. By introducing the boolean *unique* to qualify waiting_for, the unfused list is defined as follows:

<REGION>
LAST_TIME
<WAITING_FOR>, UNIQUE
<UNFUSED_LIST>

Given an unfused region R_u, the match finder organizes the search for R_u's best possible match by listing all of the regions in the other image that may be fusable with R_u. List members must pass a simple test for "fusability" that is similar to testing whether two minimum bounding rectangles intersect or not. This test for fusability is fast, but optimistic—referencing only the origins and lengths of regions' boundary traces. Regions in the image opposing R_u that satisfy this test are placed in R_u's opposing regions list. The regions in the list are ordered by quick, inexact evaluations of their matches with R_u: the ratio of fusable height to virtual height. Note that this computation does not account for fusable widths.

The implementation finds the opposing regions that pass the fusability test in a slow but easy search. There are two permanent arrays, each listing all of the regions in one of the images. They are ordered by their location: top-down, then left-to-right. Given that R_u is a region in the left image, the search begins at the top of the right

image's region array. As regions are found that pass the fusability test, they are placed in R_u's opposing regions list. The search usually terminates mid-array when a region in the array is found to be too low in the image for fusing with R_u. Because of their order in the array, the rest of the regions will be as low or lower in the image.

There is a more "distributed" way of finding fusables that is faster and more direct: connect the two spatial networks, before the fusion process begins, with horizontal pointers that link the insides of the stereo images—each region R in the left image pointing to the east-most region in the right image with which R is fusable; each region R in the right image pointing to the west-most region in the left image with which R is fusable. Using two arrays of the regions that are along the insides of the images would be a memory efficient solution for a sequential machine, since the lengths of the arrays are merely equal to the height of the image. Knowing the vertical extent of R_u, the first opposing region that is fusable can be indexed directly by line number. The other fusable regions would be found by following the horizontal neighbor relations in the spatial network.

After R_u's list of opposing regions has been formed, the match finder progresses as follows. Starting with the beginning of the list, each opposing region is considered as a possible match for R_u. Further, for each opposing region that is already bound to a match, R_u is considered an extension of it. Thus, the match evaluator is called once or twice for each fusable region opposing R_u until all have been considered. If no match is possible for R_u or the best possible match must be postponed, the match finder skips R_u on the unfused list to begin processing the next unfused region. Otherwise, a match for R_u is made and control passes to "unsettle context of new match".

Expanding existing matches with unfused regions during "find new match" is more general than during "grow active matches" for two reasons: 1) a match may be expanded with a region that is not a neighbor and 2) expanding the match may increase its local doubt.

1. When a match is grown, only a neighbor is added. Growth is limited to the current context of the match. When finding a new match for an unfused region, however, all the matches in the opposing image are candidates for expansion of R_u. As a result, a match may be expanded with a non-neighbor—fragmenting the match.

2. The match finder's condition for expanding a match with an unfused region is more relaxed than the growth condition because the match finder considers all of the unfused region's alternatives.

An important refinement of the match finder for efficiency involves *time*. During the late stages of the fusion process, the regions at the beginning of the unfused list may be repeatedly considered for matching with everything possible when only isolated contexts in the super-network are being changed using the short regions at the end of the unfused list. The match finder needs to reconsider making a match for an unfused region only in the contexts that have changed since the last attempt. By noting the times of changes and attempts to change, most redundant attempts to match unfused regions can be eliminated.

The fusion process maintains timing information using the global clock *now*, the touched_time subfield of regions, and the last_time subfield of the unfused list. Whenever a region R_u is skipped on the

unfused list because no match for R_u was found or because matching R_u is postponed, R_u's last_time is given the value of *now*. When the match finder makes a match for a region farther down the list, control passes to "unsettle context of new match". When control eventually returns to "find new match" and R_u is reconsidered for growth, R_u's touched_time and associated unfused list subfields, last_time and waiting_for, are inspected. If R_u's touched_time is more recent than its last_time or if R_u is waiting_for another region R_{au} and R_{au}'s touched_time is more recent than R_u's last_time, then R_u's last_time is reset to 0. Under these circumstances, match making attempts with R_u must be completely reconsidered. Otherwise, the match finder considers matching R_u with only the opposing regions that have a touched_time that is more recent than R_u's last_time. If a match of R_u is possible with a region recently touched by change, yet R_u is waiting_for another region R_{au}, R_u's match to R_{au} must be evaluated to determine which is best regardless of R_{au}'s touched_time.

Like the match grower, the match finder uses upper bounds for acceptable doubt to lower the total expense of match evaluations. The regions that oppose R_u are ordered so that the better candidates for matching are usually considered earlier. The results from evaluating earlier hypothetical matches of R_u are used to cut-off later hypothetical match evaluations of R_u. Like the match grower, the match finder sends upper bounds for local and total doubt to the match evaluator.

An important advantage of the ordering of R_u's opposing region list is that a last part of the list can be discarded early. If a possible best 1-to-1 match has already been found for R_u and its total doubt is less than could be possible with any of the rest on the opposing list,

further considerations are unnecessary. This happens when the non-increasing ratios of fusable height to virtual height become too doubtful.

IT HALTS.

The fusion process always terminates. This is a consequence of guaranteed progress, as expressed by the "common condition for breaking matches"—the breakable condition.

Below I argue that every change to the super-network satisfies the breakable condition, but first—why does the condition guarantee progress? The breakable condition is simply "for every match worsened or eliminated, a new high will be made". The condition is enforced by computing the new total doubts of all matches that will be affected by the change. This includes a new match and a broken match, and all their neighboring matches. The best matches of two groups are then compared: 1) the "before" total doubts of those that worsened versus 2) the "after" total doubts of those that improved. A new high for every change means progress. The lower limit on total doubt, 0, guarantees that the progress converges.

Every change to the super-network satisfies the breakable condition. Given that R_u means unmatched region, R_m means matched region, and M means match, all "changes" can be considered by case:

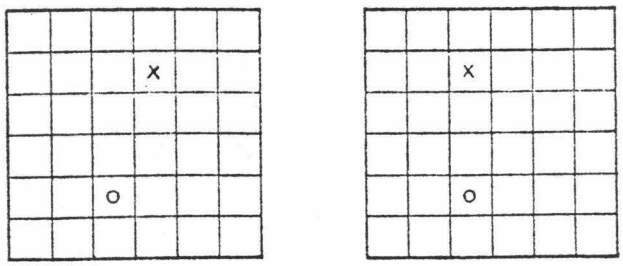
1. Find New Matches
   a. match R_u to another R_u
   b. match R_u to R_m
   c. R_u expands M
2. Grow Active Matches
   a. M absorbs R_u
   b. M absorbs R_m
3. Unsettle Context of New Match: conflict

In cases 1b, 1c, 2b, and 3 of change, the breakable condition is an explicit requirement for action. This leaves changes 1a and 2a.

Change 1a, making a completely new match, will never increase the total doubts of neighboring matches. This was made clear in III D. Feature Match Evaluation. In summary, a context of unmatched regions is as bad as the worst stereo context.

Change 2a, the last to consider, always satisfies the breakable condition because improvement of local doubt is a growth condition. As a result, no neighbors to the old parts of the match will be worsened with increased doubt. By absorbing an unfused region, the match gains a new stereo context part. Since the new part replaces no stereo context, the new neighbors will not be worsened by the change.

The determinism of the process does not mean that stereo images are always fused correctly. Consider a scene containing a checked pattern, a fly, and a spider. Assume that the image of the pattern is perfect, the fly is sitting on the plane of the pattern, and the spider is dangling in front of the pattern by an invisible silk thread. In the following picture, X marks the spider and O marks the fly.



The images of the spider and fly are the only features with unique matches. The spider and the fly vie to establish a context in the pattern. Assuming the images of the spider are larger, the spider will "get" the fly. The fusion sequence is:

1. matching large parts of the pattern is postponed
2. the spider's features are matched
3. postponed pattern parts that are neighbors to the spider are matched
4. matches of pattern parts propagate outward from the spider's context
5. the fly is matched, out of order in a maximally doubtful context

This fusion is incorrect because the pattern is not cohesive with the fly, which dooms it. It is fair, though. If the fly had been larger than the spider, the fusion would have been correct.

Although *our* vision systems may be initially fooled by the spider, a small head movement would usually unsettle the apparent scene and lead to the correct fusion.

IV. 3-SPACE FORM

The last sub-process of the system is 3-BUILD. After a stereo pair of images has been fused by EYES, 3-BUILD refines the network of matches by imposing left-to-right order and then converts the result into polygons. The polygons form a partial mold of the visual surfaces—a model of the apparent scene in the robot's functional coordinate system.

IMPOSING ORDER

In section III C, Perspective Differences, I explained that if a left-to-right order of matches in both images was not enforced, illusionary surfaces could appear. The primary sources of these illusions are surfaces that are visible in one image and occluded in the other image. Occluded features are matched to other occluded features because there is no competition.

Order is not imposed during the fusion process. Order is encouraged, however, because cohesiveness between neighboring matches is a contextual measure of match quality. Since the scope of the context that participates in the evaluation of a match is one neighbor distance, inside and outside, only matches along the borders of disorder are doubtful. For example, given the left image "1234567" and the right image "4567123", and seven matches 1-1, 2-2, 3-3, ..., there are two pairs of mutually non-cohesive neighbors: matches 7-7 and 1-1 and matches 3-3 and 4-4. Just as cohesion is a mutual relation,

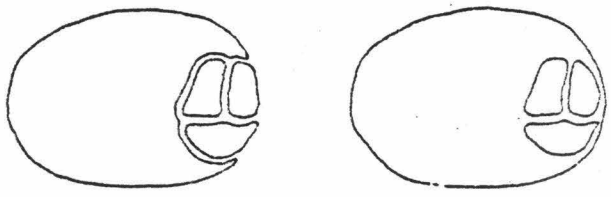
non-cohesive neighbors are mutually doubtful. Until the fusion process terminates, there is no certainty with which to impose order.

A simple algorithm for filtering out disorderly matches has four steps:

1. List all matches in the order of their total doubts, least first.
2. "Fix" the match at the front of the list.
3. Remove the match from the front of the list.
4. If the match at the front of the list is out of order with any match that has already been fixed, then go to step 3 else go to step 2.

This is approximately what I implemented. In the spider and the fly example at the end of section III E, the fly's match would be the only match not fixed because its context is maximally doubtful.

The ordering algorithm above may waste correct stereo information by breaking large matches which are only partially out of order. In the following example, only a match of the large region's eastern border would be out of order with the matches of the other regions.



Instead of disorderly regions, disorderly boundaries should be dissolved. The decision can be made on a line by line basis. In the above example, the large region is doubtful locally because the disparity on the eastern border does not change smoothly and is doubtful contextually because the eastern border is not cohesive with the neighbors. Dissolving disorderly boundaries seems to be a well defined task.

The last step of the vision process, defining polygons to model the surfaces in the apparent scene, is a direct computation—basically a well defined computer graphics task. First, stereo points are projected into the apparent scene via the camera geometry. Then, the points are connected by lines, defining boundaries in 3-Space. Last of all, the space enclosed by the boundaries is covered by polygons.

The computational basis of surface modeling is the transformation from the stereo camera space to the robot's Cartesian space, the 3-Space in which the robot functions. The origin of the robot's 3-Space is arbitrary. In the implementation, I used the point midway between the camera's focal points as the origin. The stereo coordinates of a matched point (line, $column_L$, $column_R$) or (line, column, disparity) are transformed into (X, Y, Z) coordinates via the stereo camera's geometry.

The fusable boundaries of matches define points in 3-Space. On a line by line basis, the east/west boundary of a match's region(s) in the left image fuses with the east/west boundary of the match's region(s) in the right image. If the match has multiple vertical parts, each part is considered separately. Since each match of a boundary pixel to a boundary pixel defines a point in 3-Space, a match will define a dotted

outline in 3-Space.

How is a match's dotted outline in 3-Space covered by polygons? 3-BUILD starts at the top and moves down. The first three non-linear points—such as top left, top right, and next-to-top left—define the plane for the first polygon. Successive points are included if they are in the same plane. When a point is encountered that fails the "point is in current plane" test, the current polygon is concluded and a new plane is defined. The new plane is defined by using the new point and the last two points of the polygon just concluded.

If the *nearest* spatial relation presented in Appendix 2 for hexagonal pixels had been implemented, 3-BUILD would have to be refined. The "surface scope" of a match should correspond to the contextual scope of the match. This specifically applies to neighboring insiders and to insiders that are near their outsider's boundary. If a single large region is a hole or a mound, the matches of regions inside of it define the interior contour. A random-dot stereogram, for example, consists almost entirely of near but non-touching regions inside of a large background region. Instead of covering only the interior of each dot with polygons, the dot's polygon(s) should extend outward to cohesive neighbors.

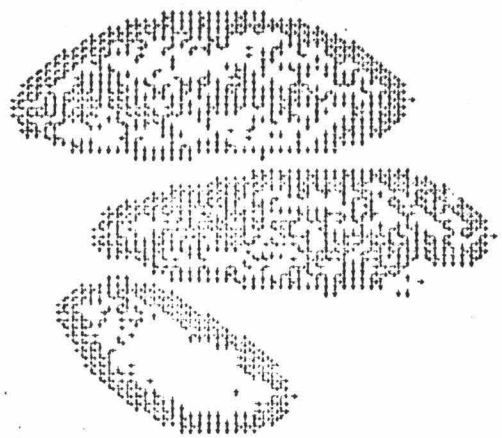
For debugging purposes, I used a hidden surface algorithm to display the results of 3-BUILD on plotting devices. The polygons defined by 3-BUILD are the input to the hidden surface algorithm. Interactively, I give a viewing position and direction, an image size, and a focal length to the program via the terminal keyboard. This

describes a specific projection of the scene onto the screen of the display. Accompaning the polygons that model the surfaces of the apparent scene, I could give the program a polygonal model of the original stereo cameras. By seeing the apparent scene along with the robot's cameras, the orientation of the apparent scene with respect to the ground and line of view is clear.

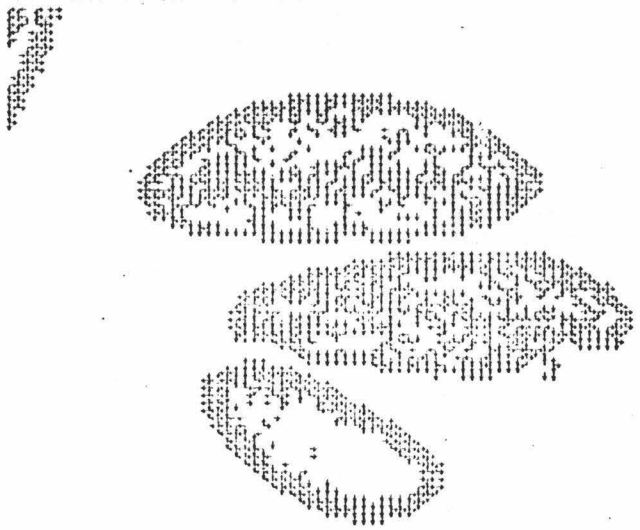
"Robot" and apparent scene

The next six pages illustrate a sample execution of the fusion process, culminated by a graphical reconstruction of the scene. The scene is simple semantically: 3 rocks. The images are complex, however, because they are fragmented into hundreds of features. The apparent scene on page 109 resulted from an incomplete fusion of the images. Note, for instance, that the bottom half of the central rock is absent.

This stereo pair of images is not exciting because the scene encompasses only 4 or 5 levels of disparity. This is revealed by the side view of the apparent scene with the robot's cameras.

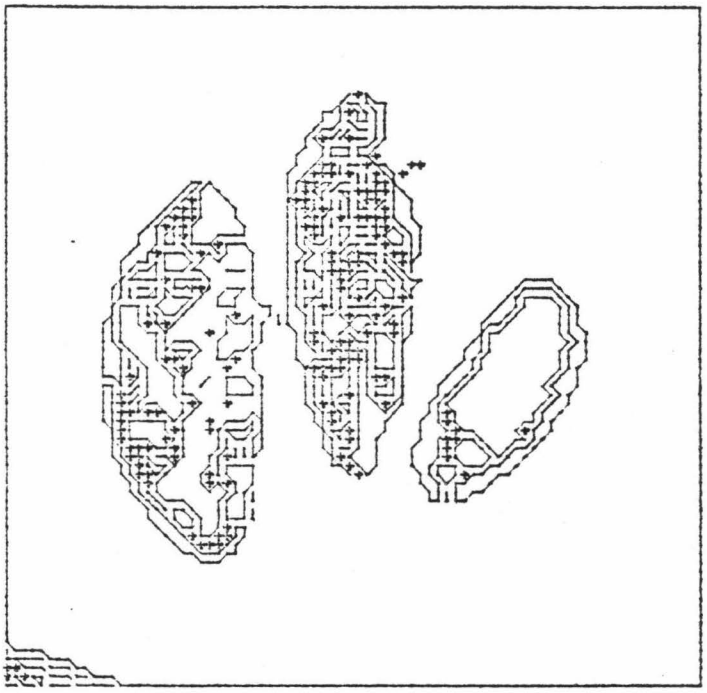


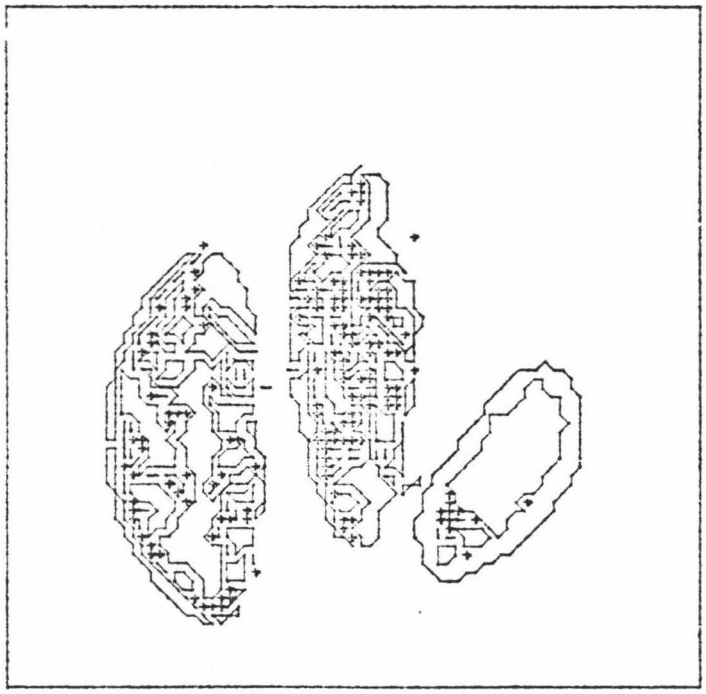
Right image



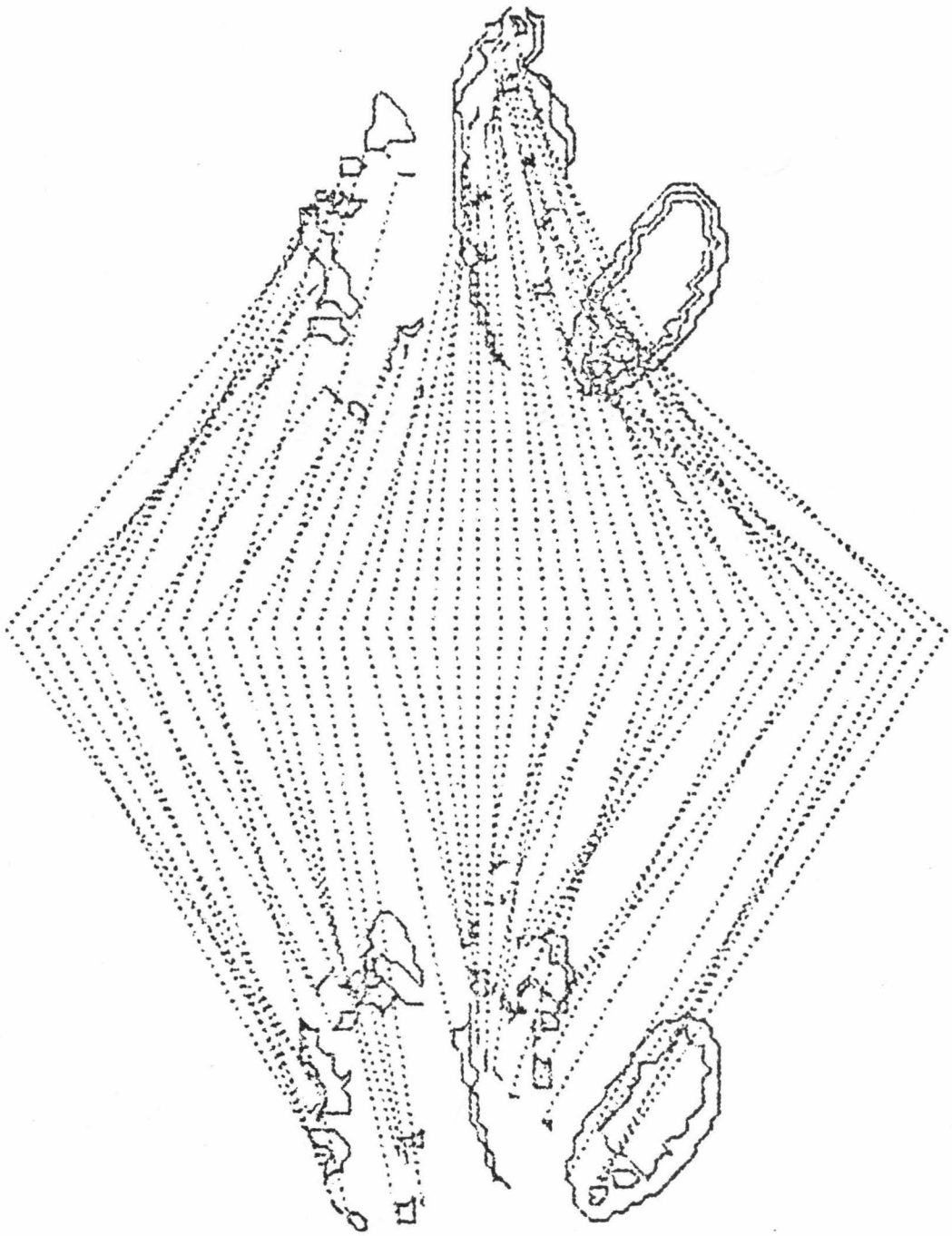
Left image

Gradient arrows

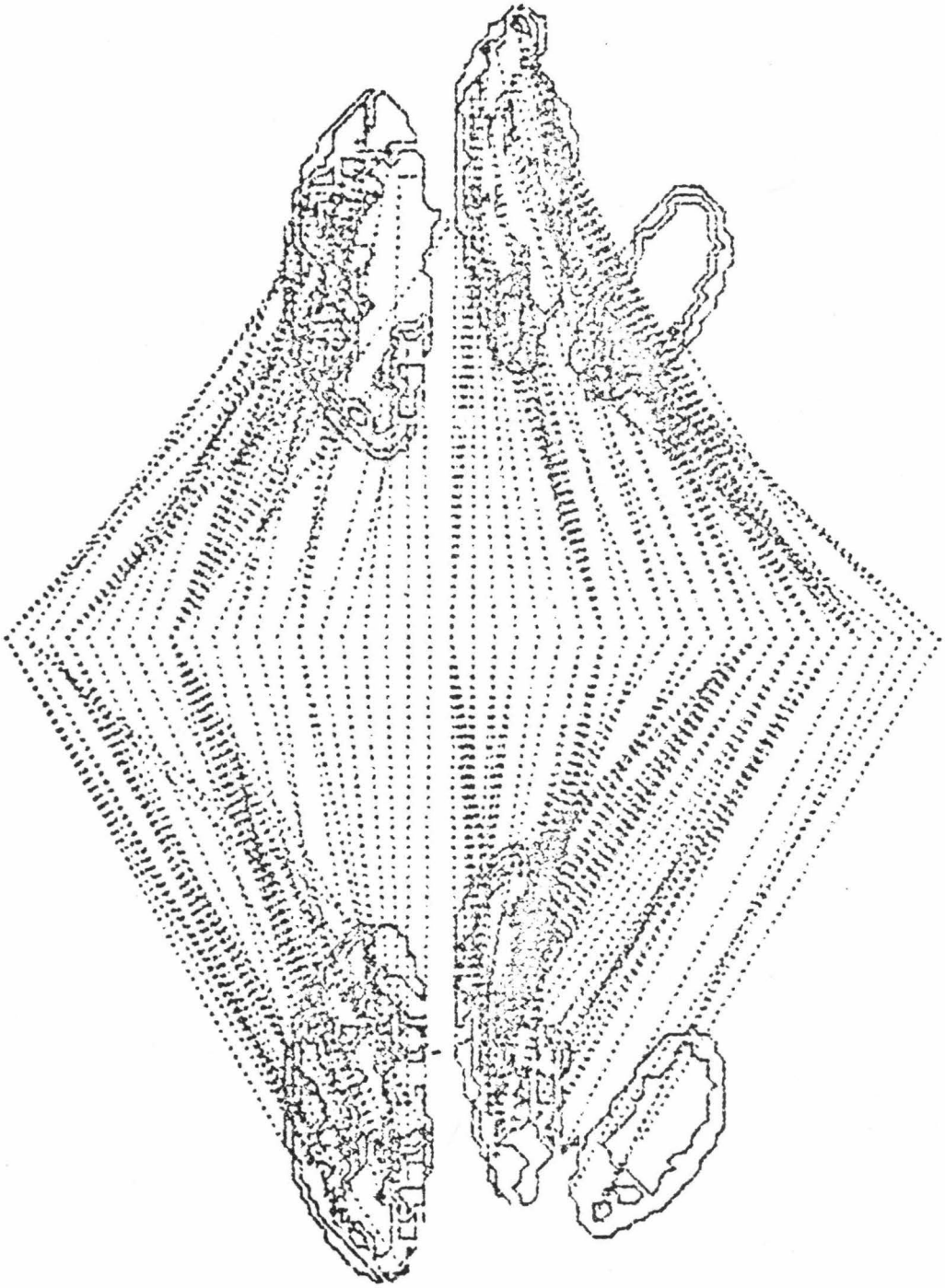




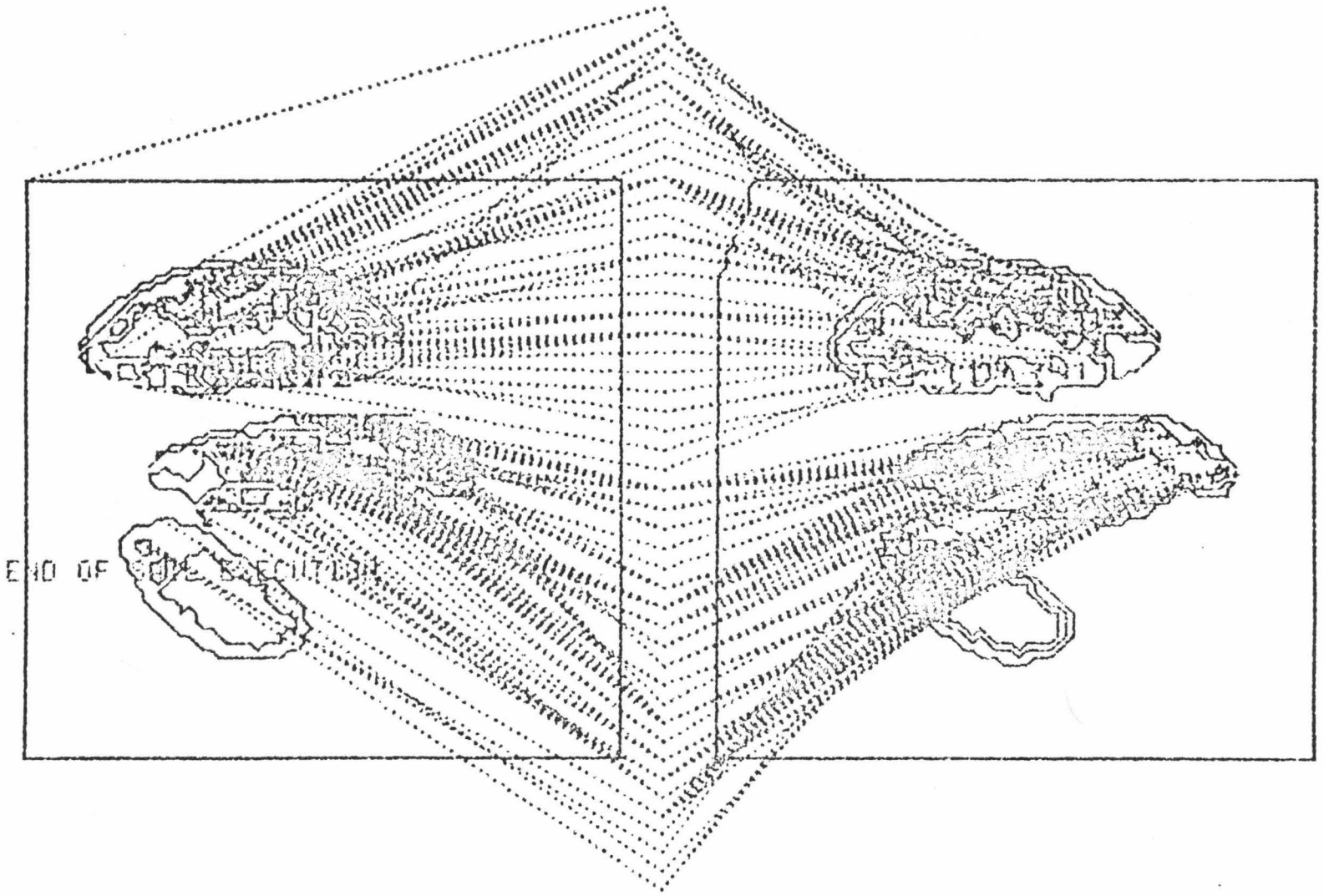
Boundaries of regions



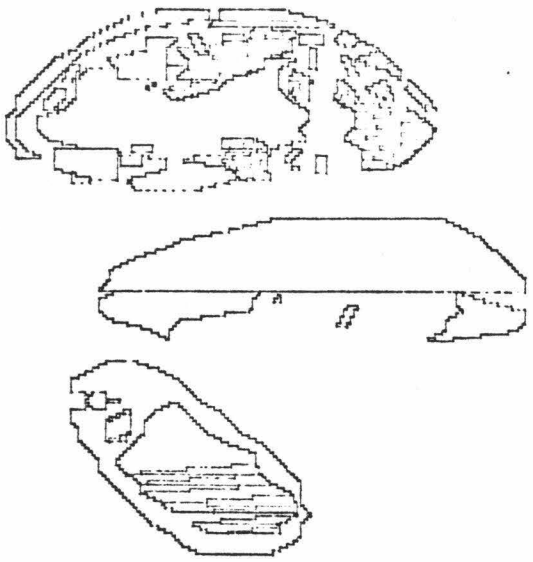
Sample 1-to-1 matches



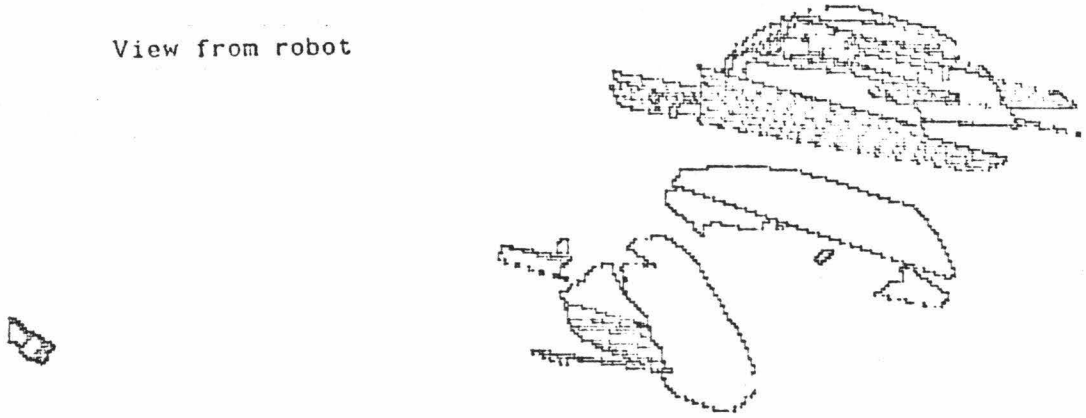
Many matches



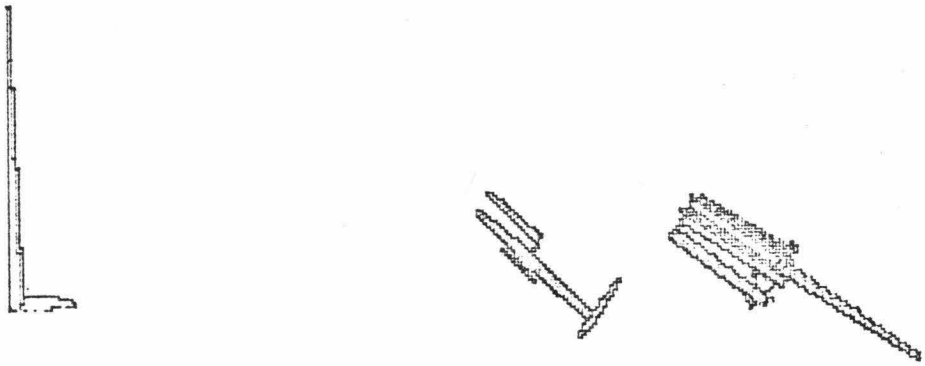
More matches



View from robot



23 degree rotation



90 degree rotation

V. CONCLUSIONS

I sought a logical understanding of stereopsis and believe I have presented a mechanical model for stereo vision at a new, nearly complete level of detail. Many robotic applications exist, but my implementation is not a marketable product for two reasons: it is inefficient and incomplete. Now as I conclude this thesis, I am confident that efficiency is the only significant problem that remains.

Within an order of magnitude, the execution time that my implementation should require on the PDP-10 given about 100x100 pixel images is:

L_EYE and R_EYE:	1 minute each
EYES:	30 minutes
3-BUILD:	2 minutes

Specific times will vary with the complexity of the images. Complexity is basically a function of the number of regions in the images—the more features, the more complex. I presented two sample stereo images in this thesis. The stereo pair illustrated in III E. Fusion Process Control Structure used approximately 15 minutes of CPU time on the PDP-10 and the stereo pair illustrated in IV. 3-Space Form used approximately 30 minutes of CPU time. These execution times should not be taken too seriously, however. Making the refinements I recommended in the body of this thesis should speed up EYES, the fusion process. As a comparison, I note that my hidden surface algorithm also takes minutes

of CPU use given an "average" scene. I worked harder to optimize that program than I did to optimize EYES, an experimental program.

A second aspect of the complexity issue, different from the number of features, involves ambiguity. In the spider and the fly example at the end of III E. Fusion Process Control Structure, there were many postponements of matches because of the ambiguity of the checked pattern. The example is artificial, however, since perfect patterns in the image are highly improbable. In this example, a match of part of the checked pattern would probably be made before the spider would be considered. This first match would then "disambiguate" the scene, directing the fusion of the rest without extensive postponements.

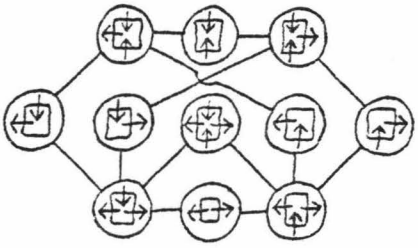
One more thesis on mechanizing stereo vision, assuming hexagonal image arrays are used, should be sufficient to demonstrate impressive results. Its description of the process will undoubtedly be more elegant and will probably be simpler.

A stereopsis algorithm with much concurrency is needed for guiding the design of a fast hardware vision system. Realizing a capability for processing stereo movies in "real time" would be a great accomplishment.

APPENDIX 1

What About the Other Cells Not in the Sphere Network?

As explained in II C. Patterns of Change, there are 33 arrow cells in the sphere network out of the possible 81 (i.e., 3^4). 1 of the 81 is the arrowless cell. This leaves 47 (i.e., 81-33-1) unaccounted for. The arrows in 25 of the sphere network's 33 cells can be reversed generating 25 of the remaining 47. These would be in the network of arrow cells produced by the negative of a sphere image. The remaining 22 (i.e., 81-33-1-25) consist of the following 11 and their arrow reversals.



Group links for non-sphere arrow cells

The bottom three cells define a horizontal bar. Similarly, if the top three cells in this network are arranged vertically—their only possible arrangement as linked neighbors—they define a vertical bar. The remaining five cells in the middle are unclassified.

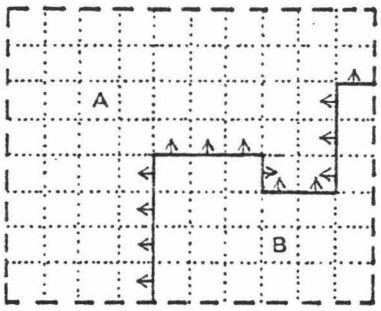
APPENDIX 2

Completing the Spatial Network

In this appendix, I outline the solution to the *neighborhood problem* by using hexagonal image arrays. I raised the problem initially in II E. Spatial Network: non-touching but *near* insiders of a larger region are not directly related in my implementation. A stereo vision system that does not make neighboring insiders readily accessible to each other will not be able to fuse random-dot stereograms.

Given a region R, R's outsiders are located when defining R's boundary trace. As the boundary is traversed, the image cells immediately on the other side of the boundary are referenced to determine the identities of their regions. These regions are R's outsiders. If an outsider's boundary touches R's boundary, then R is outside of the outsider. Otherwise, R is inside of the outsider.

For rectangular arrays, neighboring insiders could be related by generalizing the outside relation. Instead of looking only one image cell beyond R's boundary, a multi-cell search could be made perpendicular to the boundary until a neighbor's boundary is encountered. Consider the following image window. A and B are regions. The visible boundary belongs to B, so B is inside of A.



Search directions for *nearest*

In the spatial network, the new outside—or nearest—relation must have the additional information: separation distance. When the distance is 1, the nearest relation is simply the old outside relation.

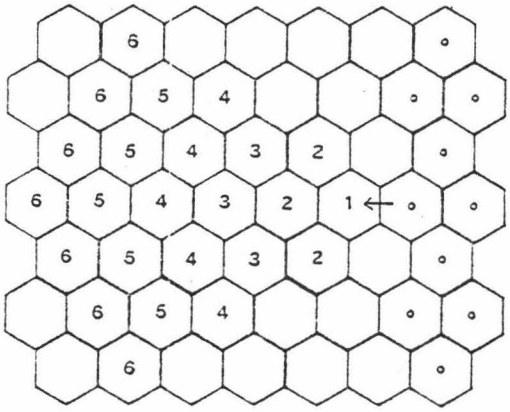
This solution to the neighborhood problem in rectangular arrays is imperfect because the scope of a neighborhood will not encompass the entire image. Diagonal spatial relations are lost. A solution must be possible, but it will undoubtedly be less elegant than the general solution for hexagonal images.

By using hexagonal image arrays, a number of problems and a lot of computer code disappear. The advantages of hexagons over rectangles are as follows:

- regions never intersect
- neighboring pixels are all equidistant
- chain links are all the same length
- if oriented correctly, there are no vertical neighbors

The only real disadvantage of hexagonal images is that hexagonal cameras, video equipment, etc. are not yet available in the market.

The search path for nearest neighbors of hexagonal regions can be defined so that the entire image will be covered, given an isolated region having any shape. In the following illustration, the hex pattern is oriented so that there are no vertical neighbors—only east, west, northeast, northwest, southeast, and southwest.



The dotted pixels are members of a single region. The numbered pixels denote the search order for a nearest boundary. When found, it is to be related to the region boundary across which is an arrow.

By starting with the hexagonal array, I think the criteria for match evaluation could be neatly reconstructed. Instead of beginning with the assertion that each line of a feature has equal weight because only vertical boundaries are important, a very similar conclusion could be derived by initially giving each pixel equal weight. Further, I think the spatial parts of local and contextual match quality would become fully integrated, particularly regarding the smoothness of cohesion.

APPENDIX 3

Evaluating Features with Multiple Vertical Parts

In III D. Feature Match Evaluation, features with multiple vertical boundaries were ignored for simplicity. Actually, all regions have "multiple" boundaries: an east side and a west side. Features like the letter "M" may produce even more vertical boundaries, however. The problem left unresolved in section III D is "how are the multiple parts distributed for fusing?".

When matched, the vertical parts of an M-shaped feature are assigned to the vertical parts of another feature so that the parts' left-to-right order is preserved. Matching the left image's west-most part to the right image's west-most part and matching the left image's east-most part to the right image's east-most part, within fusable limits, are the first steps.

If two disjoint areas in one image must be matched to a single area in the other image, the area between the two disjoint areas in one image becomes an exclusively virtual area. The exclusively virtual area is locally doubtful and its contextual doubt will depend on occlusion excuses.

The distribution problem that remains concerns the matching of multiple internal parts to opposing multiple internal parts. Since features are matched on a line by line basis, the problem reduces to matching stereo pairs of multiple intervals. The solution I implemented considers, a line at a time, all combinations and chooses the least doubtful locally, counting only the doubts of exclusively virtual areas.

The algorithm is recursive and uses the results of previous recursions to cut-off future ones.

This solution is not completely satisfying because the distribution of vertical parts that the implementation chooses may not be the best—that is, the resulting match may not have least total doubt. By evaluating all possible distributions using *all* of the match criteria, the best would be found. The computation is more expensive, however, and the combinatorics of "all possible distributions" could be a problem. From a global view of the process, the time required to match a feature with many vertical parts should be proportional to the time required to match the vertical parts, given that they are separate features.

Consider a comb's stereo images, for example, oriented so that the comb's teeth point up or down in the images. Although an image of a comb will probably not be a single feature with teeth distinct, the possibility must be considered.

There is an advantage to having an "intact" comb over a "fragmented" comb: the left-to-right ordering. The left-to-right ordering significantly reduces the number of match distributions to evaluate. Individual features are not constrained to a specific ordering during the fusion process. Neighboring matches that are not mutually cohesive are merely contextually doubtful along their interface.

A complete solution to the problem of matching features with many vertical parts so that the best is found using all of the match criteria may be possible. By enriching the single distribution criterion implemented—least amount of exclusively virtual area—with horizontal

and vertical disparity smoothness, a "best" distribution of vertical parts can be found directly. Then, the total doubt of this first match possibility can be used to cut-off most of the alternatives. This should work for intact combs, even if a tooth is missing in one of the stereo images.

Last of all, the fusion process presented in the body of this thesis must be refined in two ways in order to accommodate matches of features with multiple vertical boundaries.

1. Refinement to match evaluation: Instead of each line of the match having potential local/contextual doubt of 10/10 before being normalized for height, each horizontal *interval* of the match has local/contextual doubt of 10/10 before being normalized by the total number of intervals.

2. Refinement to process control structure: When a match with multiple vertical boundaries is grown—expanded by an additional feature—the breakable condition must be satisfied. This is necessary because the addition may cause the matches of vertical parts to be redistributed, thereby changing interfaces to old neighbors. In fact, the match grower should attempt to expand these matches with null features when the matches are activated, thereby allowing their internal vertical parts to be redistributed in response to changes in their contexts.

REFERENCES

1. Barrow, H.G. and Tenenbaum, J.M., "MSYS: A System for Reasoning about Scenes", Stanford Research Institute, Technical Note 121, April 1976.

2. Bejczy, A. K., "Effect of Hand-Based Sensors on Manipulator Control Performance", *Mechanism and Machine Theory*, Vol 12, Pergamon Press, Great Britian, 1977.

3. Gregory, R. and Wallace, J., "Recovery from Early Blindness: A Case Study", *PERCEPTION Selected readings in Science and Phenomenology*, Tibbetts, P. (Editor), Quadrangle Books, Chicago, 1969.

4. Hannah, M., "Computer Matching of Areas in Stereo Images", Ph.D. Thesis, Stanford University, July 1974, NTIS index: AD-786 720, (quote from first line of abstract).

5. Horn, B., "Obtaining Shape from Shading Information", *The Psychology of Computer Vision*, Winston, P. (Editor), McGraw-Hill, New York, 1975.

6. Hueckel, M., "An Operator Which Locates Edges in Digitized Pictures", AI Memo 105, Stanford University, October 1969.

7. Joyce, J., *Finnegans Wake*, The Viking Press, New York, 1972, pg 52.

8. Julesz, B., *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago, 1971.

9. Marr, D. and Poggio, T., "Cooperative Computation of Stereo Disparity", SCIENCE, Vol 194, October 1976, pp 283-287.

10. Marr, D., "Early Processing of Visual Information", AI Memo 340, MIT, December 1975.

11. Newman, W. and Sproull, R., *Principles of Interactive Computer Graphics*, McGraw-Hill Book Company, New York, 1973.

12. Ohlander, R., "Analysis of Natural Scenes", Carnegie-Mellon University, April 1975.

13. Piaget, J., *The Mechanics of Perception*, Basic Books, Inc., New York, 1969.

14. Reiser, J. (Editor), *SAIL*, AI Memo 289, Stanford University, August 1976.

15. Thompson, A., "The Navigation System of the JPL Robot", JPL Publication 77-20, Caltech, 1977.

16. Tuong-Phong, B., "Illumination for Computer-Generated Images", Ph.D. Thesis, University of Utah, July 1973.

17. Udupa, S., "Collision Detection and Avoidance in Computer Controlled Manipulators", Ph.D. Thesis, Caltech, September 1976.

18. Waltz, D., "Understanding Line Drawings of Scenes with Shadows", *The Psychology of Computer Vision*, Winston, P. (Editor), McGraw-Hill, New York, 1975.

19. Yakimovsky, Y., "Scene Analysis Using a Semantic Base for Region Growing", Ph.D. Thesis, Stanford University, June 1973.

20. Author unknown.