Spring 5-31-1990

A stereo vision technique based on the multi-positioned camera criterion

Jie Bao
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

A Stereo Vision Technique Based on the Multi-Positioned Camera Criterion

by

Jie Bao

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1990

APPROVAL SHEET

Title of Thesis: A STEREO VISION TECHNIQUE
BASED ON THE MULTI-POSITIONED CAMERA CRITERION

Name of Candidate: JIE BAO
MASTER OF SCIENCE
IN ELECTRICAL ENGINEERING, 1990

Abstract and Thesis Approval:

Dr. Ali N. Akansu — Date

Dr. Nirwan Ansari — Date

Dr. Zoran Siveski — Date

VITA

Jie Bao

Education

1988-1990	New Jersey Institute of Technology MSEE
1981-1986	University of Science and Technology of China BSEE

Experience

1986-1987	Automation Institution, Academia Sinica Electrical Engineer
1987-1988	Autowell, LTD, Hongkong Electrical Engineer

ABSTRACT

Title of Thesis: A Stereo Vision Technique
Based on the Multi-Positioned Camera Criterion
Author: Jie Bao, Master of Science in Electrical Engineering, 1990

Thesis directed by: Dr. Ali N. Akansu

A modified feature based stereo vision technique is described in this thesis. The technique uses the curve segments as the feature primitives in the matching process. The local characteristics of the curve-segments are extracted by the Generalized Hough Transform.

A set of images of a scene, which are taken by a multi-positioned camera satisfying the parallelism criterion, are first filtered by the Laplacian of a Gaussian operator in different widths, i.e. coarse to fine channels. At each channel, the Generalized Hough Transform is applied to the curve-segments in each image. The curve position, the curve-length, the curve centroid, the average gradient of the curve-segment and the R-table are used as the local features in representing the distinctive characteristics of the curve-segment. These features of all the curve-segments in an image are used as the constraints to find the corresponding curve-segments in the different images. The epipolar constraint on the centroid of the curve-segment is used to limit the search window in the images. Since the multi-images of one view are used, there exist more information about the scene than the two relational images criterion. Its performance compares to the other matching techniques, for example, the point matching, or twin image matching that the mismatching and the calculation are greatly reduced. Although the algorithm is not feasible for the realization of the real-time implementation of stereo vision, it is a more economic way of finding the depth of an object or a view.

ACKNOWLEDGMENT

Many people have assisted me do my thesis work. I am very grateful to my graduate advisor Dr. Ali N. Akansu whose inspiration and guidance benefitted me significantly; without his support this work could not have been finished.

I would also like to express my appreciation to my friends at NJIT, who gave me their assistance in many ways. I feel particularly grateful to Dwan-Yen Fang, Huan-Chung Lee, Yi-Zhong Chen and Bacon Yi who helped me so much during my study at NJIT. And I am greatly indebted to my uncle Andrew Bau and my cousins in USA who have supported me over my first year in America.

At last I would thank my wife's encouragement and understanding during my study in America, and also my dear family which supported me during these years.

Dedicated to

My Wife Hong-Hong and My Family

Contents

Approval Sheet

Vita

Abstract

Acknowledgment

Contents

List of Figures

Chapter 1 Introduction **1**

1.1 Camera Geometry 3

1.2 Synopsis 7

Chapter 2 Edge Detection **10**

2.1 Edge 10

2.2 Choice of Filter 11

2.3 Window Size and Orientation 14

2.4 Coarse-to-Fine Strategy 16

Chapter 3 Curve Tracking **19**

3.1 Sequential Segmentation Methods 19
3.2 Feature Primitive 20
3.3 Raster Tracking 21

Chapter 4 Generalized Hough Transform **25**

4.1 Introduction 25
4.2 The Hough Transform for Analytic Curves 26
4.3 The Hough Transform for Non-Analytic Curves 27

Chapter 5 Matching Process **32**

5.1 Introduction 32
5.2 Multi-Channel Graph Matching 33
5.3 Details of the Algorithm 34

Chapter 6 Discussion **39**

6.1 Introduction 39
6.2 Zero-Crossings 39
6.3 Curve Segment and Hough Transform 41
6.4 Matching Process 42

Appendix A Original Simulation Program Listing **45**

A.1 Definitions DEF.H and DMASK.H 45
A.1.1 DEF.H 45
A.1.2 DMASK.H 46
A.2 Main Simulation Program 46
A.2.1 VZ0VIS.C 46
A.2.2 VZ0VIS1.C 48
A.3 Fast Fourier Transform VZ1FFT.C 49

A.4 Function Group VZ2SUB.C 52
A.5 Curve Tracking VZ3SEG.C 58
A.6 Generalized Hough Transform VZ4HOU.C 59
A.7 Matching Process VZ5SIM.C 60

Appendix B Simulation Results **62**

References **102**

List of Figures

1.1 Parallel Axes Method 4

1.2 False Targets Problem 6

2.1 Spatial and Directional Factors Interact in the Definition of Edges 13

2.2 The Primal Sketch Operator 15

4.1 Geometry for Generalized Hough Transform 27

4.2 Construction of **R**-table 28

5.1 Flow Chart for Matching Algorithm 36

5.2 Flow Chart for Node Assignment 37

B.1 Original Image of Tab 63

B.2 Original Image of Jack 64

B.3 Original Image of Screwdriver 65

B.4 Zero-Crossings of Tab at $\sigma^2 = 30$ 66

B.5 Orientation of Tab Zero-Crossings at $\sigma^2 = 30$ 67

B.6 Curve-Segmnets of Tab at $\sigma^2 = 30$ 68

B.7 Disparity Picture of Tab at $\sigma^2 = 30$ 69

B.8 Zero-Crossings of Jack at $\sigma^2 = 30$ 70

B.9 Orientation of Jack Zero-Crossings at $\sigma^2 = 30$ 71

B.10 Curve-Segmnets of Jack at $\sigma^2 = 30$ 72

B.11 Disparity Picture of Jack at $\sigma^2 = 30$ 73
B.12 Zero-Crossings of Screwdriver at $\sigma^2 = 30$ 74
B.13 Orientation of Screwdriver Zero-Crossings at $\sigma^2 = 30$ 75
B.14 Curve-Segmnets of Screwdriver at $\sigma^2 = 30$ 76
B.15 Disparity Picture of Screwdriver at $\sigma^2 = 30$ 77
B.16 Zero-Crossings of Tab at $\sigma^2 = 20$ 78
B.17 Orientation of Tab Zero-Crossings at $\sigma^2 = 20$ 79
B.18 Curve-Segmnets of Tab at $\sigma^2 = 20$ 80
B.19 Disparity Picture of Tab at $\sigma^2 = 20$ 81
B.20 Zero-Crossings of Jack at $\sigma^2 = 20$ 82
B.21 Orientation of Jack Zero-Crossings at $\sigma^2 = 20$ 83
B.22 Curve-Segmnets of Jack at $\sigma^2 = 20$ 84
B.23 Disparity Picture of Jack at $\sigma^2 = 20$ 85
B.24 Zero-Crossings of Screwdriver at $\sigma^2 = 20$ 86
B.25 Orientation of Screwdriver Zero-Crossings at $\sigma^2 = 20$ 87
B.26 Curve-Segmnets of Screwdriver at $\sigma^2 = 20$ 88
B.27 Disparity Picture of Screwdriver at $\sigma^2 = 20$ 89
B.28 Zero-Crossings of Tab at $\sigma^2 = 10$ 90
B.29 Orientation of Tab Zero-Crossings at $\sigma^2 = 10$ 91
B.30 Curve-Segmnets of Tab at $\sigma^2 = 10$ 92
B.31 Disparity Picture of Tab at $\sigma^2 = 10$ 93
B.32 Zero-Crossings of Jack at $\sigma^2 = 10$ 94
B.33 Orientation of Jack Zero-Crossings at $\sigma^2 = 10$ 95
B.34 Curve-Segmnets of Jack at $\sigma^2 = 10$ 96
B.35 Disparity Picture of Jack at $\sigma^2 = 10$ 97
B.36 Zero-Crossings of Screwdriver at $\sigma^2 = 10$ 98

B.37 Orientation of Screwdriver Zero-Crossings at $\sigma^2 = 10$ 99

B.38 Curve-Segmnets of Screwdriver at $\sigma^2 = 10$ 100

B.39 Disparity Picture of Screwdriver at $\sigma^2 = 10$ 101

Chapter 1

Introduction

Stereo vision systems have a large number of applications such as inspection of objects in three dimensions for defect identification or verification of product specification, recognition of objects in three-dimensionional models of a robot work cell for robot arm control. In all these stereo vision applications, three-dimensional information about the environment is essential. Depth information is important for the control system as well as for object modelling and recognition.

Methods of obtaining depth information have been presented by many researchers. One of these methods uses camera system and multi-images are obtained by moving the camera from one position to another. The distances between the camera positions can be known from the camera calibration and robot arm calibration. The details of camera calibration and robot arm calibration can be found in [1], [2], [3] and [4].

The development of a stereo system involves image acquisition and preprocessing, feature extraction, registration (with recovery of camera parameters), matching, depth-from-disparity and surface reconstruction. The major problem in stereo vision techniques is to find the corresponding points in the stereo images. The corresponding points are the projections of a single point in the

three-dimensional space. The difference of the projections of a point into their respective images is called disparity. Disparity is a function of both the position of the point in the scene and the position, the orientation, and the physical characteristics of the multi-positioned camera. There is no complete solution to this problem, but constraints can be used to reduce the search. In this thesis, we will focus on the feature extraction and the matching algorithm.

The camera geometry in any stereo vision system is very important. It is possible to limit the search space for matching image points to one dimension. In this thesis, the positions of the camera which is mounted on the robot arm are set up that their focal axes are parallel and the distances between the camera positions are known as baselines, b, which is fixed as shown in Figure 1. This is known as the parallel axis geometry.

The proposed technique is a feature based matching technique where curve-segments are used as the feature primitives, combined with the multi-positioned camera geometry. Curve-segments are used because the figural continuity constraint is inherently embedded in the matching primitives and ambiguity in finding matches for the curve-segments is much less than the point matching algorithms. The local characteristics of a curve segment is represented by the curve-length, the average gradient along the curve-segment, and the R-table of the Generalized Hough Transform of the curve-segment. An instance of the curve-segment in other images is sought by using the local characteristics of the curve-segments, and the constraints based on the graph matching technique. In other matching techniques, the process of eliminating false matches tends to be complex and slow, and many errors remain. Because many of the accepted matches may be spurious, not corresponding to the physical structure in the scene. A multi-channel graph matching technique is also used here to reduce

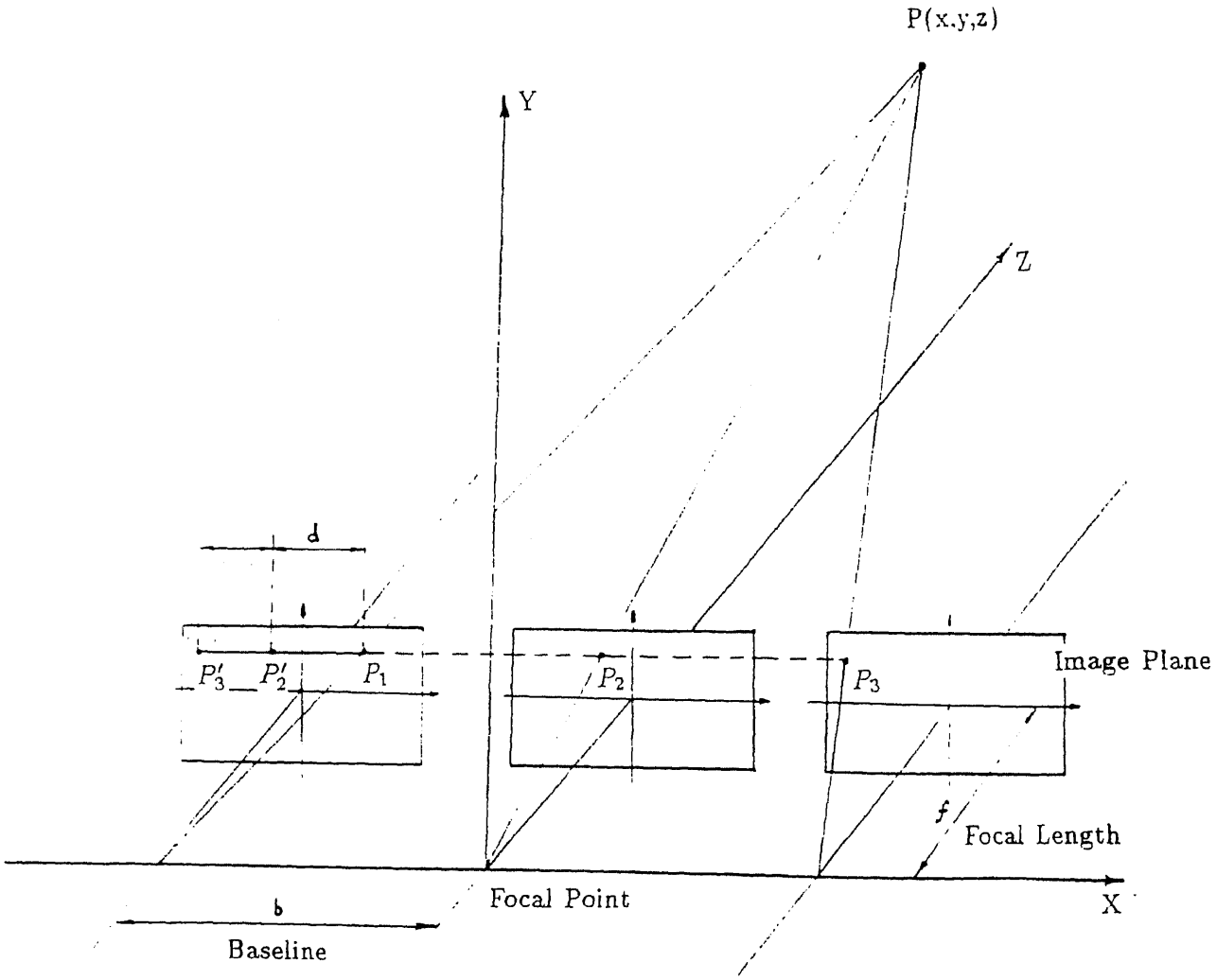
the ambiguity of the disparity in the two image pairs.

1.1 Camera Geometry

In our setup, a SONY CCD camera is mounted on a platform with a fixed baseline and parallel focal axes. Hereafter we will refer to this as the parallel axes method. The camera moves along the baseline so that the horizontal scan lines of both camera positions are parallel to the baseline and all the flow of disparities are horizontal and unidirectional on one of the image planes. In this case, the fixation point may be assumed to lie at infinity. Any point in the three dimensional space, together with the centers of the projection of the multi-positioned camera system, defines a line called the epipolar line. In the parallel axis geometry the epipolar lines are parallel to the scan lines. Thus the search for finding the corresponding points is unidirectional as shown in Figure 1.

Consider a point $P(x, y, z)$ in the world coordinate system that corresponds to points P_1, P_2 and P_3 in the left, center and right image coordinate planes respectively. We assume that the imaging system is linear, does not require any correction. The camera calibration has been studied and those results are used here [4]. The distances $P_2'P_1$ and $P_3'P_2'$, etc., where P_2' is the transformed location of P_2 and P_3' is the transformed location of P_3, are known as disparities. It can be shown that the distance Z is inversely proportional to the each sub-disparity, which is the difference in the positions of the two corresponding points in two different images of the neighboring camera positions. The points which are closer to the camera position have a larger disparity than the points which are

Figure 1.1: Parallel axes method, the positions of the camera are set up such that their focal axes are parallel and the line joining the focal centers is perpendicular to it.



farther away from the camera position.

$$Z = b\frac{f}{d} \tag{1.1}$$

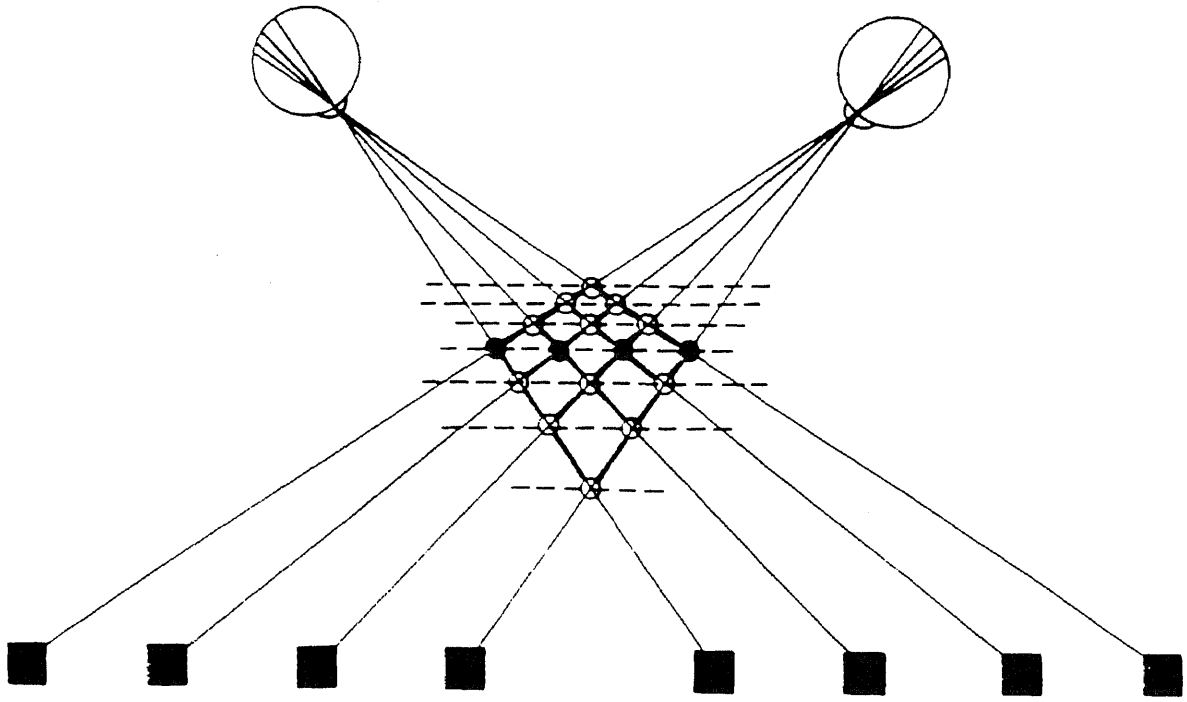
where $d = P_i'P_{i-1} \quad i > 1$ and the disparity is a function of the point position in the scene and the position, the orientation and the physical characteristics of the camera respectively. Therefore once the location of the target points P_i and P_{i-1} are identified, the distance Z_i can be calculated.

From last formula, it can be seen that we can easily find the depth of the point if there is only one point in the scene using the camera setup. But if there are more than one point in the scene, you can not find the right depth of one particular point easily by using this method. This introduces the major problem of stereopsis. The task of identifying corresponding locations in the images is a difficult one because of what is called the *false target problem*. If you can not find the correct corresponding points in the different images, the depth you detect will be the depth of one false target. This is shown in Figure 2.

There are four parallel points in the scene of Fig. 2. We take two pictures separately from the left and right camera positions. There are four points in each picture. We want to find the corresponding point in the right picture to a point in the left image. If we choose the corresponding points randomly in the right image, we'll get three false targets. So there exist twelve false target points. The probability we have the correct depth information of the four points is only 0.25. If we have two gray level intensity images, it's almost impossible for us to find the correct depth information of the whole image. Therefore reducing the false targets is the main and hardest work of stereo vision.

To answer this question, additional information is needed to help to decide

Figure 1.2: False targets problem. The 16 possible matches indicated by the circles. Only those indicated by the filled circles are actually perceived.



which matches are correct by constraining them in some manner. The only way to do this is to examine the basis in the physical world to make a correspondence between the images. Marr and Poggio describe two physical constraints which are relevant to the stereo process [7]:

(1) A given point on a physical surface has a unique position in space at any one time.

(2) Matter is cohesive, it is separated into objects, and the surfaces of objects are generally such that the changes in the surfaces are very small compared with their distance from the viewer.

The computational problem for the stereo process may be summarized as follows. First, for each view of the scene, construct a description of the elements that are to be matched. The second step consists of solving this corresponding problem, determining which descriptor from one view matches to which descriptor from the other view.

1.2 Synopsis

The thesis consists of six chapters.

In Chapter 1, a summary of the thesis is included. Edge detector, tracking algorithm, Hough transform and matching process are introduced. The camera geometry is also explained in this chapter.

Chapter 2 provides a discussion of the theory of edge detection. The coarse-to-fine strategy of the MPG algorithm is explained.

A tracking algorithm is given in Chapter 3. It is used to segment the boundaries of the objects in the scene into short curve-segment. These curve-segments are used as the feature primitives in the matching process.

Chapter 4 introduces the Generalized Hough Transform. It is used to identify an instance of each curve-segment of one image to the other images.

In Chapter 5, a multi-channel graph matching process is presented. Relaxation matching techniques are used to obtain a global match between the multi-position image curve-segments.

The last chapter, Chapter 6 includes the discussion of the experimental results and conclusions. At last the outlined future work of this stereo vision system is presented.

The thesis also includes an appendix which includes all the source programs of the system simulations.

Chapter 2

Edge Detection

It is impossible to identify surface locations unambiguously from the images themselves at all points. It has been shown in the Chapter 1. A smooth featureless surface will be of no meaning to this process, because the image intensities will be indistinguishable over this surface. Any surface containing scratches, texture or other markings will, however, give rise to locally sharp changes in reflectance which can be used to define surface points. Since only certain features of a surface will be well defined in the images, it is claimed (Marr 1974, Marr and Poggio 1979) that the computation of disparity takes place by comparing symbolic descriptions of those features in the images.

2.1 Edge

It would be pleasant if one could design a single filter to extract all the features from a scene. Unfortunately, such intensity changes take place over a wide range of scales (Marr, 1976). For example, if we look at individual picture elements (pixels), we can find intensities changing from pixel to pixel. However, such changes occur at too small scales to be of interest to us. Most of the edges in the real world are sharp edges, and the associated intensity function will be composed of a few steep changes over a small number of pixels. At the

same time, other edges, such as shading edges, are spatially extended, and the intensity function will increase slowly over a large number of pixels. Moreover, these different types of intensity changes are not distinct in the image; one can have high contrast edges superimposed on a spatially extended shading edges.

As a consequence of this wide range of intensity changes, one can not hope to find a filter which will be simultaneously optimal for all scales. The findings of Campbell and Robson (1968), concerning the existence of separate spatial-frequency channels in the human visual system, suggest that one should seek a method of dealing with the changes occurring at different scales separately. As a consequence, Marr and Hildreth [7] suggest that one first take some local average of intensities at several resolutions, and then detect the changes in intensities that occur at each resolution. They determine both the optimal smoothing filter and a method for detecting intensity changes at a given scale.

2.2 Choice of Filter

The optimal smoothing filter must satisfy two physical constraints. First, the filter is intended to reduce the range of scales over which intensity changes take place. This suggests that the filter's spectrum should be band-limited. Second, the aspects of an object which give rise to intensity changes are all spatially localized. This suggests that the filter should perform an average over a small localized portion of the image. These two requirements are in conflict and are related by a type of uncertainty principle. It has long been known that under these circumstances, the optimal filter for minimizing band-width in space and frequency is the Gaussian. Thus, the image I is initially convolved with a two

dimensional Gaussian operator G.

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-x^2/2\sigma^2) \tag{2.1}$$

In two dimensions,

$$G(r) = \frac{\pi\sigma^2}{2} \exp(-r^2/2\sigma^2) \tag{2.2}$$

For each channel, the size of the operator G will vary. The filter G thus provides the trade-off between the conflicting requirements.

Wherever an intensity change occurs, there will be a corresponding peak in the first directional derivative, or equivalently, a zero-crossing in the second directional derivative of intensity. In fact, we may define an intensity change in this way, so that the task of detecting these changes can be reduced to that of finding the zero-crossings of the second derivative D^2 of intensity, in the appropriate direction.

That is to say, we seek the zero-crossings in

$$f(x,y) = D^2[G(r) * I(x,y)] \tag{2.3}$$

where $I(x,y)$ is the image, and $*$ is the convolution operator. By the derivative rule for convolutions,

$$f(x,y) = D^2G * I(x,y) \tag{2.4}$$

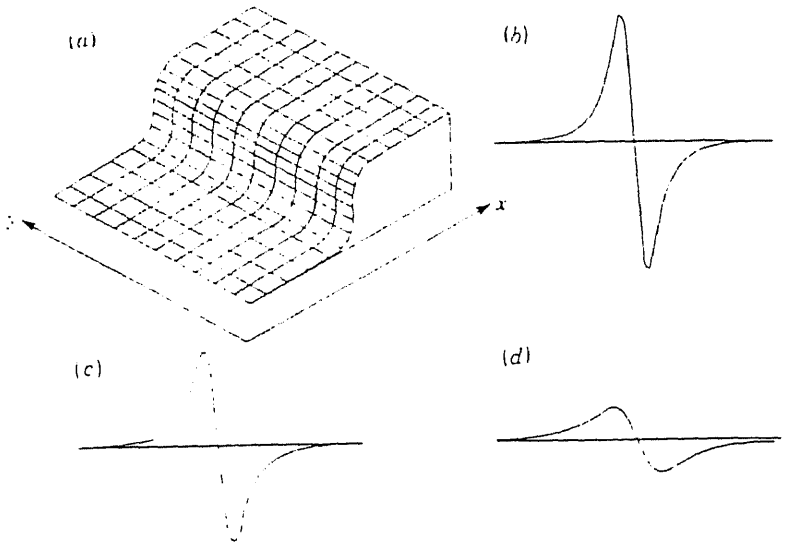
We can write the operator D^2G in one dimension

$$D^2G(x) = -\frac{1}{\sigma^3\sqrt{2\pi}} \exp(-x^2/2\sigma^2) \tag{2.5}$$

This operator is roughly band pass, therefore it examines a portion of the image spectrum.

The final thing left to do is to determine the direction in which the directional derivative must be taken. A number of practical considerations (Marr and

Figure 2.1: Spatial and directional factors interact in the definition of edges. (a) shows an intensity, and (b) (c) and (d) show values of the second directional derivatives near the origin at various orientations across the change. In (b), the derivatives are taken parallel to the x-axis, and in (c) and (d), at 30° and 60° to it. There is a zero-crossings line up along the y-axis, this is the direction that is chosen. In this example, it is also the direction that maximizes the slope of the second derivative.



Hildreth, 1979) led the initial operators not be directional in nature. If this is the case, then the operator to be used is the Laplacian, since it is the only non-directional linear second derivative operator, ∇^2. It was shown (Marr and Hildreth 1979) that provided two simple conditions on the intensity function in the neighborhood of an edge are satisfied, the zero-crossings of the second directional derivative taken perpendicular to an edge will coincide with the zero-crossings of the Laplacian along that edge. Then the form of the operator is given be:

$$\nabla^2 G(r) = \frac{r^2 - 2\sigma^2}{\sigma^4} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} \tag{2.6}$$

This is a rotationally symmetric function.

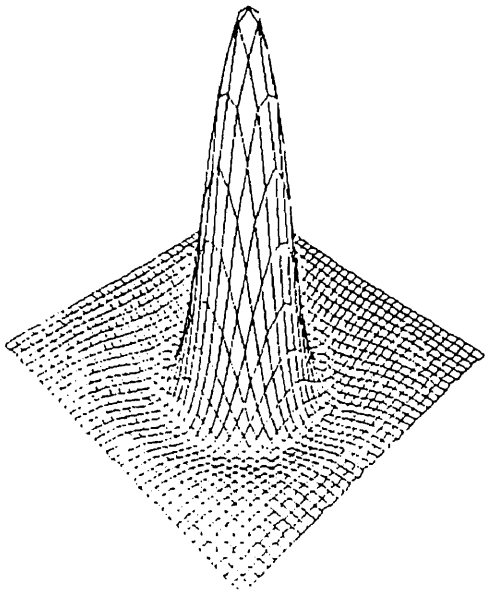
2.3 Window Size and Orientation

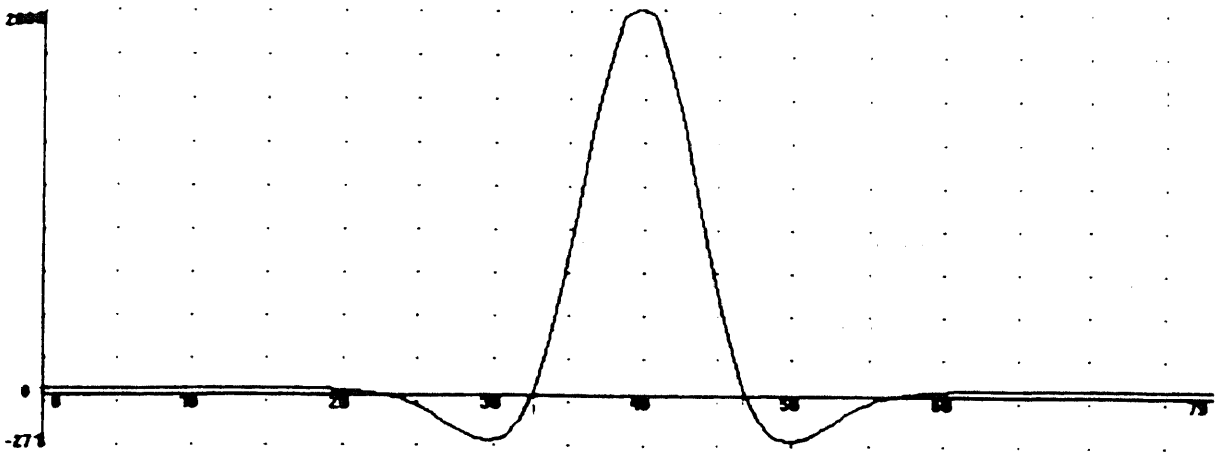
The cross-section of the Laplacian of a Gaussian operator is shown in Figure 2.2. Note that its central panel width, denoted by w, and defined as the width of the central negative region, is given by

$$w_{2-d} = 2\sqrt{2}\sigma \tag{2.7}$$

Zero-crossings are obtained by scanning along each processed image line and column, locating pairs of adjacent elements of opposite signs. Some results are shown in the Appendix. The orientation θ_{xy} of the zero-crossings is calculated from the local gradients in the X and Y direction of the zero-crossings, which are detected from the profile of the convolved image. The orientation at each edge pixel is needed in the curve tracking algorithm and in the Generalized Hough Transform evaluation. The local gradients in the X and Y direction are

Figure 2.2: The primal sketch operator. It shows a cross-section of the rotationally symmetric two-dimensional operator. The size of the operator is determined by the values of w_{2-d}. The operator size is limited to a window size of $1.8w_{2-d}$ because the magnitude of the coefficients falling outside this window is very small.





calculated by the following Sobel gradient operators:

$$\delta Y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$\delta X = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.9}$$

Then the orientation is calculated by,

$$\theta_{xy} = \tan^{-1} \frac{\delta Y}{\delta X} \tag{2.10}$$

The gradient is calculated as,

$$m_{xy} = \sqrt{\delta^2 Y + \delta^2 X} \tag{2.11}$$

2.4 Coarse-to-Fine Strategy

It is observed that the physical phenomena that the intensity changes in the images are spatially localized. Since these changes produce zero-crossings in the filtered images, it follows that if a discernible zero-crossings are present in a channel with central width w_0, there should be a corresponding zero-crossing at the same spatial location in the channels with the central widths $w > w_0$. If this is true at some wavelength $w' > w_0$, it will be for one or two reasons: Either (a) two or more local intensity changes are being averaged together in the larger channel. Situations of this kind can be recognized by small channels but not by larger ones. Or (b) two independent physical phenomena are operating to produce intensity changes in the same region of the image but at different scales. Situations of this kind can be recognized if the zero-crossings in the larger channels are displaced relative to those in the smaller ones. If they have the correct positions and orientations, the locations of the zero-crossings may

not contain enough information to separate the two physical phenomena, but, in practice, this situation will be rare.

Provided that the zero-crossings from adjacent independent channels coincide, they can be taken together. If they do not, they probably arise from distinct surfaces or physical phenomena, maybe the shadow or so. It follows that the minimum number of channels required is two, and the two channels are reasonably separated in the frequency domain, and their zero-crossings agree, the combined zero-crossings can be taken to indicate the presence of an edge in the image.

For the stereo images, the zero-crossings representations are obtained from the coarsest filters with central width w_0 first. We suppose that the root disparity in a region of the image is d_0, which is an arbitrarily estimated value. For a zero-crossing, in one image at (x, y), the search for a matching in the other image is constrained to the region

$$\{(x', y) | x + d_0 - w_0 \leq x' \leq x + d_0 + w_0\} \tag{2.12}$$

Finally, once this matching has been performed for the coarsest channel, the average disparity obtained can be used to realign the images, and the process can be repeated at the next finer scale. Since the density of zero-crossings increases as the size of the filter is decreased, this coarse-to-fine strategy allows the matching of very dense image descriptions with greatly reduced false target problems by using coarser resolution matching to drive the alignment process.

Details of the matching process are given in the following chapters.

Chapter 3

Curve Tracking

3.1 Sequential Segmentation Methods

There are several segmentation methods for edge and curve tracking. Some of them can be performed on pictures "in parallel ", i.e., at all points simultaneously. The processing of any point in the image is independent of the other points. They can be implemented very efficiently on a suitable parallel computer. In this thesis, we consider one kind of sequential segmentation methods. In this method, the criterion for accepting a point as part of an object can depend on the information obtained from earlier processing of other points, and in particular, on the natures and locations of the points already accepted as parts of the objects. However, in the parallel approach, the same amount of computation must be performed at every point of the image, since our only basis for accepting or rejecting a given point is the result of its own computation. When using the sequential approach, we can use simple and inexpensive computations to detect possible object points. Once some such points have been detected, more complex computations can be used to track the objects, but only at the points that have been already detected. The squential segmentation computation is relatively cheap, since they only need to detect some points of any object

while rejecting non-object points.

Sequential segmentation methods have a potential advantage over parallel methods, with respect to their computational cost on a conventional, sequential computer. And sequential methods are very flexible, and can be defined in many different ways.

3.2 Feature Primitive

We choose curve-segments as our matching feature primitives because of their three desirable characteristics. First, a curve-segment can still be identified even if it is partially occluded. Also, the figural continuity constraint is automatically satisfied by using curve-segments. Finally, the problem of finding a unique match for a curve-segment is much less ambiguous than for a point target and should produce fewer mismatches.

Another advantage of using curve-segments as a feature primitive over edges is that a graph can be formed for each image to represent the local properties of the curve-segments as well as the relational (structural) properties of the curve-segments. Consequently, a graph matching technique (relaxation) may be used to obtain a global match between the curve-segments of the images. It is also important to note that the centroids of the curve-segments should satisfy the epipolar constraint imposed on them by the camera's geometry.

To identify each curve-segment, the Generalized Hough Transform representation of the curve will be evaluated in the next chapter. This represents the iconic property of the curve-segment and is used to identify the instance of the same curve-segment in different images.

3.3 Raster Tracking

The detection and tracking criterion can involve local properties rather than gray level. In this thesis, we use gradient as the reference. Note that the gradient not only a degree of contrast, but also a direction (of highest rate of change of gray level at the point); in tracking,one could look for successive points along the perpendicular direction, which should be the direction along the curve being tracked.

Suppose that the objects to be extracted from the given picture are thin, dark, continuous curves whose slopes never differ greatly from 90°. This is true for our case because we use the zero-crossing profile as our search reference. The zero-crossing profile is purely black and white picture. And for the edges, the zero-crossings are continuous. The current tracking algorithm segments the boundaries of the objects, i.e., zero-crossings in a scene into short curve-segments.

We scan the gradient magnitude and orientation pictures of the image row by row in the same manner as TV raster scanning, according to the zero-crossing profile. Especially, in each row of the gradient we accept any point whose gradient level exceeds some relatively high threshold as the starting point for the tracked curve-segment. The relatively high level is an experimental value. The optimal relatively high levels are different for different pictures . Once a point (x, y) on the yth row has been accepted as the starting point of a curve-segment, we accept any neighbor of it on the $(y-1)$th row, i.e., we accept any of the points $(x-1, y-1)$, $(x, y-1)$, and $(x+1, y-1)$, provided that it has gradient level above the pre-defined low level and its orientation does not differ by more than a pre-defined threshold from the previously tracked point

(x, y). The low level and the angle threshold are also experimental values. They are defined experimentally. If the difference in the orientation between the two tracked points exceed the angle threshold, or if the magnitude of the newly tracked point dips below the low level, the curve is broken off. This process is recursively continued until the curve is broken off. If we used the high level alone, or the low level alone, the curves would not be extracted correctly; but when we combine the thresholds, in conjunction with row-by-row tracking, we are able to extract the curves.

The proceeding remarks can be summarized in the following generalized raster tracking algorithm.

1. On the first row (or line of the raster) accept all points that meet detection criterion. Take each point to be the initial point of a curve C_i that is to be tracked.

2. On any current row other than the first row.

   (i) For each curve C_i currently being tracked, apply the appropriate tracking criterion to the points in its acceptance region; adjoin the resulting accepted points to C_i. We recall that this criterion may depend on the distances and directions of these points from the end of C_i or from some curve that extends C_i. If no new points are accepted into C_i, the tracking of C_i is terminated. Note that a curve C_i may branch into two or more curves, in this case we must track them all; or two or more curves may merge into a single curve, in this case, we need only track that one from there on.

   (ii) In addition, apply the detection criterion to points that do not belong to any accepting region; if any point meet this criterion, take it to

be initial point of new curve C_j.

3. When the bottom row is reached, the tracking process is complete.

In the tracking algorithm, each curve-segment is labelled by a number, the location of all the edge points forming the curve-segment are stored. Total length of the curve-segment is also kept in the memory buffer, but relatively very short curve-segments are discarded. It may change for different channels. And the centroids of each curve-segment are calculated as,

$$\overline{c_x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.1}$$

$$\overline{c_y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.2}$$

where N is the length of the curve-segment and (x_i, y_i) the coordinates of the edge pixels. The above information is then used in the Generalized Hough transform algorithm to generate the distinctive R-table of each curve. A set of curve-segments is thus obtained for the left, central and right images.

Chapter 4

Generalized Hough Transform

4.1 Introduction

To identify each curve-segment in one of the triple stereo images we must extract and use some distinctive features of the curve. We use the Generalized Hough Transform of the curve as a distinct property of the curve-segment. The Generalized Hough Transform is used because it groups all the information about the individual zero-crossings on the curve into a single table which uniquely represents that curve-segment. The Generalized Hough Transform is very robust in finding a match for a curve-segment even when some part of the curve is occluded, or when edge pixels are noisy and point-to-point matching is not possible.

The Generalized Hough Transform algorithm uses edge information to define a mapping from the orientation of an edge point to a reference point of the shape. The reference point may be thought of as the origin of a local coordinate system for the shape. In this thesis, we use the centroid of the curve-segment as the reference of the curve-segment. Then there is an easy way of computing a measure which rates how well points in the image are likely to be the origins of the specified shape. A feature of this work is that it will work when the

boundary is disconnected due to noise or occlusions. This is generally not true for other strategies which track edge segments.

4.2 The Hough Transform for Analytic Curves

We consider analytic curves of the form $f(\mathbf{x}, \mathbf{a}) = 0$ where $\mathbf{x}$ is an image point and $\mathbf{a}$ is a parameter vector. If we use directional information associated with the edge, this reduces the parameter locus to a line. What happens is the equation

$$\frac{df}{dx}(\mathbf{x}, \mathbf{a}) = 0 \tag{4.1}$$

It introduces a tern dy/dx which is known as

$$\frac{dy}{dx} = \tan\left[\phi(\mathbf{x}) - \frac{\pi}{2}\right] \tag{4.2}$$

where $\phi(\mathbf{x})$ is the gradient direction. This suggest the following algorithm.

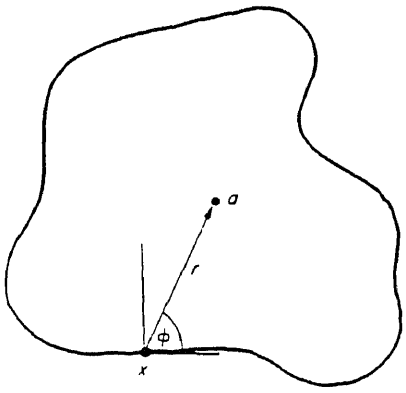
For a specific curve $f(\mathbf{x}, \mathbf{a}) = 0$ with parameter vector $\mathbf{a}$, form an array $A(\mathbf{a})$, initially set to zero. This array is called an accumulator array. Then for each edge pixel $\mathbf{x}$, compute all $\mathbf{a}$ such that $f(\mathbf{x}, \mathbf{a}) = 0$ and $\partial f(\mathbf{x}, \mathbf{a})/\partial \mathbf{x} = 0$ and increment the corresponding accumulator array entries:

$$A(\mathbf{a}) += 1 \tag{4.3}$$

After each edge pixel $\mathbf{x}$ has been considered, local maxima in the array A corresponds to curves of f in the image.

If only the equation $f(\mathbf{x}, \mathbf{a}) = 0$ is used, the cost of the computation is exponential in the number of parameters minus one.

Figure 4.1: Geometry for Generalized Hough Transform



4.3 The Hough Transform for Non-Analytic Curves

To generalize the Hough algorithm to non-analytic curves we define the following parameters for a generalized shape:

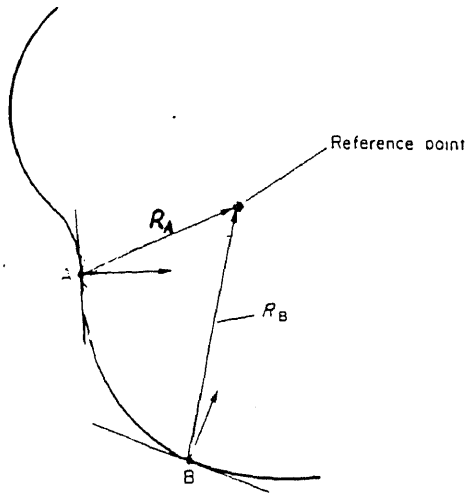
$$\mathrm{x} = \{\mathrm{a}, \mathrm{s}, \theta\} \tag{4.4}$$

where $\mathrm{a} = (x_r, y_r)$ is a reference origin for the shape, θ is an orientation, and $\mathrm{s} = (s_x, s_y)$ describes two orthogonal scale factors.

The key to generalizing the Hough algorithm to arbitrary shape is the use of directional information. Directional information, besides making the algorithm faster, also greatly improves the accuracy.

Consider an arbitrary shape, for each point x on the boundary with gradient

Figure 4.2: Construction of **R**-table



direction ϕ, we increment a point $\mathbf{a} = \mathbf{x} + \mathbf{r}$, where $\mathbf{a}$ is the reference point and the $\mathbf{r}$ is the distance between the point $\mathbf{x}$ and the reference point $\mathbf{a}$, which varies in an arbitrary way. Then the R-table is easily constructed by examining the boundary points of the shape. The construction of the table is accomplished as follows.

Algorithm for constructing a R-table. Choose a reference point $\mathbf{a}$ for the shape. For each boundary point $\mathbf{x}$, compute the gradient direction $\phi(\mathbf{x})$ and $\mathbf{r} = \mathbf{a} - \mathbf{x}$. Store $\mathbf{r}$ as a function of ϕ. Notice that the representation of R-table is vector-valued and, in general, an index ϕ may have many values of $\mathbf{r}$. The **R**-table is used to detect instances of the shape in an image in the following manner.

For each edge pixel $\mathbf{x}$ in the image, increment all the corresponding points

$\mathbf{x} + \mathbf{r}$ in the accumulator array A where $\mathbf{r}$ is a table entry indexed by ϕ, i.e., $\mathbf{r}(\phi)$. Maxima in A corresponds to possible instances of the shape.

If we use the strategy of incrementing the accumulator array by unity, then the contents of the accumulator array are approximately proportional to the perimeter of the shape that is detectable in the image. This strategy is biased toward finding shapes where a large portion of the perimeter is detectable. Several different incrementation strategies are available, depending on the different quality of image data. If shorter, very prominent parts of the perimeter are detected, as might be the case in partially occluded objects, then an alternative strategy of incrementing by the gradient modulus value might be more successful, i.e.,

$$A(\mathbf{a}) = A(\mathbf{a}) + g(\mathbf{x}) \tag{4.5}$$

Of course the two strategies can be combined, e.g.,

$$A(\mathbf{a}) = A(\mathbf{a}) + g(\mathbf{x}) + c \tag{4.6}$$

where c is a constant.

Another possibility is the use of local curvature information in the incrementaion function. Using this strategy, neighboring edge pixels are examined to calculate approximate curvature, K. This requires a more complicated operator than the edge operator we have considered, and complicates the table. Now along with each value of r the corresponding values of curvature must be stored. Then the incrementation weights informative high local curvature edge pixels as follows:

$$A(\mathbf{a}) = A(\mathbf{a}) + K \tag{4.7}$$

In this thesis, the table is constructed by calculating the orientation θ_{xy} of each pixel (e_x^i, e_y^i) of the current curve-segment E^i and the vector distance be-

tween the centroid $\overline{c^i} = (c_x^i, c_y^i)$ of the curve-segment and the edge pixel location. This distance R^i is given by $(r_x^i, r_y^i) = (c_x^i - e_x^i, c_y^i - e_y^i)$. In the **R**-table, these distances (r_x^i, r_y^i) are listed as a function of θ_{xy}. This table is used to detect instances of the same curve segment in the other image, with the additional constraint that the difference in the locations of the matching centroids is only a horizontal shift. This difference in locations is a result of the parallel axis camera geometry. A curve-segment in one image is said to be matched with a curve-segment in the other images if their R-tables are approximately the same.

Chapter 5

Matching Process

5.1 Introduction

The tracking algorithm produces a set of curves with distinct local characteristics, stored in the R-tables of the curves. The curve stereo matching problem may now be considered as a point matching problem between a set of points from one of the triple images and a set of points from the other images. Since the number of centroids is much less than the number of edge pixels, the matching is easier and faster. Also sophisticated matching techniques exist in which the relationships among the curve centroids may be used to obtain a global match between a set centroids in the triple stereo images.

A multi-channel technique is proposed to obtain a global match between curve-segments in one of the images and those in the other images. In this technique, three graphs are formed in each channel where the graphs represent the structural relationships and the local properties of the centroids of the images respectively. The curve-segments from different images which have the same or similar properties are matched ones.

5.2 Multi-Channel Graph Matching

The input to the matching algorithm is sets of curve-segments extracted from the images. Each curve-segment is identified by a number, the location of its centroid, its R-table representing the properties of the curve-segment, its curve-length and the location of each edge pixel belonging to the curve. Using the centroids and the R-table of the curve-segments, a relational graph can be formed for each image. The nodes of the graph represent the location of the centroids of the curve-segments and the arcs represent the relationship between the centroids. The R-table of the curve-segment

Let graphs $L(N_l, P_l, E_l, \sigma)$, $M(N_m, P_m, E_m, \sigma)$ and $R(N_r, P_r, E_r, \sigma)$ represent symbolically the left, the central and right images respectively where N represents the number of nodes (curve-segments), P a set of local properties of the nodes, E a set of relations between the nodes, and σ representing the channel. In our matching algorithm, the R-table of each curve-segment is the local property of the corresponding node.

For instance, if by a similarity measure the local properties P_l^i of a node N_l^i in graph L approximately match the local properties P_r^j of node N_m^j in graph M, then this pair of nodes (N_l^i, N_m^j) is said to form a node assignment, provided the epipolar constraint on the centroids is also approximately satisfied. The measure of local similarity for node assignment in our matching algorithm is then given by the ratio

$$S(N_l^i, N_m^j) = \frac{A^j}{L_l^i} \tag{5.1}$$

where A^j represents the Hough accumulator value (number of edge pixels that are matched) obtained when the R-table of the curve-segment in E_l^i the left image is compared with the R-table of the curve-segment E_r^j in the central

image and L_l^i represents the curve length of E_l^i.

5.3 Details of the Algorithm

Starting at the coarsest channel for each curve-segment E_l^i in the left image a matching curve-segment E_m^j is sought in the central image in order to form a node assignment. Due to the camera's geometry, the location of the centroid $C_l^i = (c_x^i, c_y^i)$ of the left curve-segment is displaced by a disparity value in the central image. Ideally, the disparity value is simply a horizontal shift, i.e., the centroids should be on the same epipolar line. However, due to the geometrical distortion, imperfections in the tracking algorithm, and partial occlusion, the centroids of the corresponding curves may not be exactly on the same epipolar line. Nevertheless, for most of the corresponding curve-segments their centroids will just fall within a few scan lines apart. The horizontal displacement between the centroid of the curve-segment in the left image and that of the corresponding curve-segment in the central image is called the centroid disparity.

For instance, to find the node assignment for a given curve-segment E_l^i in the left image, all nodes (centroids) in the central image that are within a 2-D search window, which size is $W_\sigma \times 15$ by 24, around the point $(c_x^i, c_y^i + \overline{d^i})$ are considered as candidates. Here, (c_x^i, c_y^i) is the location of the node of the curve-segment in the left image and $\overline{d^i}$ is the average disparity around the curve-segment which is obtained from the previous channel disparity calculation. For the coarsest channel, the value is an eliminated one.

The R-table of each candidate is compared with that of the curve-segment in the left image. Whenever a pair of nodes with a node assignment score above a threshold which is also an experimental value, we accept the curve-segment E_m^j in the central image.

To compare the R-table of the one curve-segment with that of another curve-segment in the other image, each θ^i_{xy} entry of the R-table of curve-segment a is compared with all the θ^j_{xy} entries of the R-table of curve-segment b in the other image. If the orientations are approximately the same, i.e., their orientation difference $|\theta^i_{xy} - \theta^j_{xy}|$ is below the threshold 10°, which we defined in the tracking algorithm, and $r^i_x = r^j_x$, $r^i_y = r^j_y$, then the Hough accumulator is incremented by one. Let the value A^j represents the Hough peak obtained from the curve-segment j. Then the local node assignment score is calculated by

$$S(N^i_a, N^j_b) = \frac{A^j}{L^a_i} \tag{5.2}$$

Repeating the above steps, we try to find node assignments for the curve-segment E^j_m in the central image in the right image within the window ($W_\sigma \times 15$ by 24) around the point $(c^j_x, c^j_y + \overline{d^j})$. If we are not able to find a match, we go back to the previous step and try to find another node assignment. Once we obtain a match with E^k_r, we use the curve-segment E^k_r in right image to calculate the node assignment with the E^i_l in the left image. If the assignment is below the threshold, we also have to go back to the previous matching steps. This criterion restricts the false targets significantly. If the assignment is above the threshold, we say the three curve-segment in the triple stereo images are matched. Disparities are calculated for the matched curve-segments.

Then we go back to match the next curve-segment in the left image. This procedure won't stop until all the curve-segment are matched.

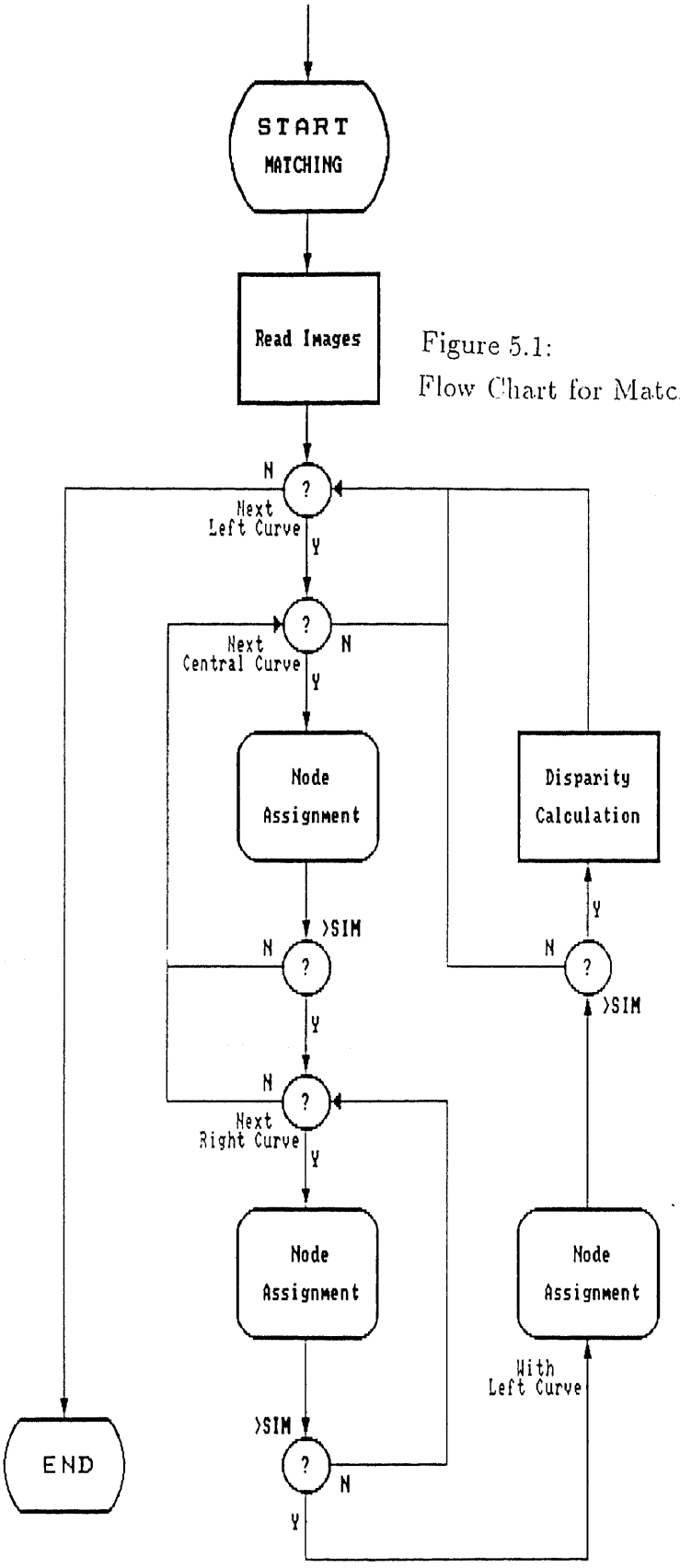


Figure 5.1:
Flow Chart for Matching Algorithm

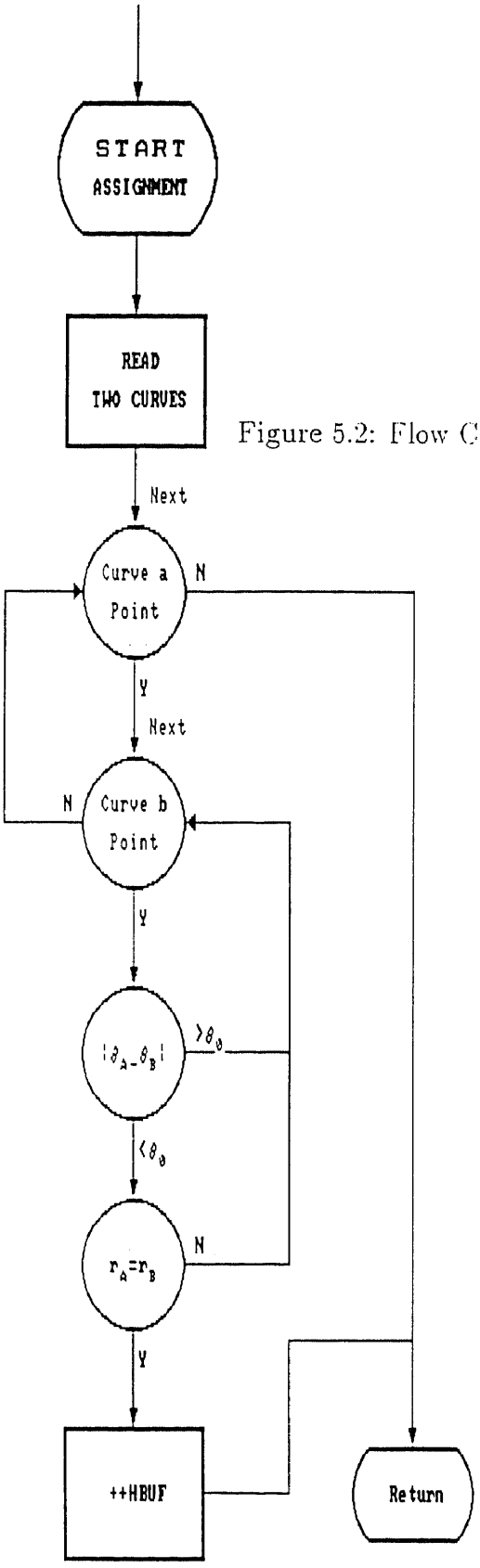


Figure 5.2: Flow Chart for Node Assignment

Chapter 6

Discussion

6.1 Introduction

This presented stereo vision technique has been implemented on Sun Workstation. Some results are discussed here. The stereo images were taken by a SONY CCD camera which is mounted on a platform. Moving the camera horizontally, we obtain a set of related stereo images. In the following examples, three pictures were taken for each object. Each picture is of resolution 512×480 and 8 bit gray level.

Main steps of the simulation algorithm:

- Zero-Crossing Extraction by a Gaussian of Laplacian Operator.
- Curve Tracking According to the Gradient Information of the Zero-Crossings.
- Generalized Hough Transform against the Curve Segments of Each Image.
- Matching Process Based on the Multi-Images.

6.2 Zero-Crossings

The zero-crossings were extracted after convolving the images with $\nabla^2_\sigma G$ for three σ^2 values between 10 to 30. This range of σ^2 does not represent the actual

size of the channels in the Human Visual System, but is adequate for experimental demonstration of the algorithm. Actually we use FFT to do the convolution instead of convolution using a mask or a window. There is an obvious advantage of this. As we increase the variance of the Gaussian filter, the convolution window will be larger. It will result of losing a lot of information on the margin pixels of the image. The bigger the variance is, the more information we are losing. FFT will avoid this kind of defection happening. Zero-crossings were detected where there were positive to negative, or negative to positive changes in the array processed by the Gaussian of Laplacian operator. They were marked as 255 in the zero-crossing array, others zero. Figure B.1 to Figure B.3 represent the stereo images of three objects, tab, jack, srewdriver. Figure B.4, Figure B.8, Figure B.12, Figure B.16, Figure B.20, Figure B.24, Figure B.28, Figure B.32, Figure B.36 show the zero-crossings of the objeacts respectively at several different σ^2 values, 30, 20, 10.

The magnitude and orientation of the gradient of the zero-crossings were obtained by convolving the Sobel operator with the original images according to the zero-crossing file of each image. We scan the zero-crossing array row by row, then column by column. Whenever we meet a positive-negative change in the zero-crossing array, we use the X-direction and Y-direction Sobel filters to convolve the corresponding pixel in the original image array. With the two values, we are able to calculate the magnitude and the orientation of gradient of the edge pixels using the equation 2.10 and 2.11. These information were used in the tracking algorithm and the Generalized Hough Transform. The orientations of the zero-crossings are shown in Figure B.5, Figure B.9, Figure B.13, Figure B.17, Figure B.21, Figure B.25, Figure B.29, Figure B.34, Figure B.38. As the low acuity of the printout, we can not distinguish the gray level

differences in the orientation pictures. However, you can easily tell the gray level changes on the screen. We translate the dynamic range of the orientation and the magnitude of the gradient to $0 \sim 255$. Because the magnitude of the gradient of the noise is very low, we are even not able to see them on the screen. Here no magnitude pictures of the gradient are shown in the thesis.

6.3 Curve Segment and Hough Transform

The details of the curve tracking algorithm were introduced in Chapter 3. Figure B.6, Figure B.10, Figure 14, Figure B.18, Figure B.22, Figure 26, Figure B.30, Figure B.34, Figure 38 show the curve-segments extracted from the triple stereo images of each object respectively. It is seen from the pictures that most centroids, which is represented by the crossmarks in the pictures, are approximately on the same horizontal line for different pictures. There is a small vertical disparity between the pictures due to the camera misalignment. Because imperfect lighting, we lose some details of the edges of the objects.

In the tracking algorithm, we defined the tracking high threshold of the magnitude of the gradient as 64 and the low 32. And the orientation difference threshold was defined as 10°. We are able to update this threshold to $3° \sim 6°$ which will also work perfectly in the algorithm for these particular pictures. Choosing the gradient magnitude thresholds is very important. So far it can only be chosen by experiments. Different thresholds will produce completely different results. Sometimes you are even not able to detect any curve segments. Sometimes you obtain a lot of noise in the pictures. More time is needed on the realization on the sophisticated tracking algorithm, the simulated technique in the thesis is not always effective and sometimes it will produce broken curve-segments. It may be caused by the lighting and/or the threshold choosing.

To identify each curve-segment, the Generalized Hough Transform representation of the curve is evaluated. This represents the iconic property of the curve-segment and is used to identify the instance of the corresponding curve-segments in the other two images. Other image properties such as curvature or slope density function could have been used to represent the iconic properties of the edge pixels. The Generalized Hough Transform can easily be extended to include other properties of the curve, such as the average gray level, gradient, curvature, color, end point location of the curve-segment, or the type of junctions terminating the curve-segment. In the thesis, we use the distance between the centroid of the curve-segment and the edge pixel location and the orientation of the edge pixels to construct the R-table.

6.4 Matching Process

The tracking algorithm produces a set of curves with distinct local characteristics, stored in the R-tables of the curves for different pictures. This curve stereo matching problem is considered as a point matching problem among the sets of nodes or centroids from different pictures. Since the number of centroids is much less than the number of edge pixels, the matching is easier and faster. Especially, because the multi-positioned camera criterion was used, sophisticated matching technique will also simplified. Figure B.7, Figure 11, Figure 15, Figure B.19, Figure 23, Figure 27, Figure B.31, Figure 35, Figure 39 show the disparity images of the "triple" stereo images. All the curve-segment are perfectly matched.

In the algorithm, the size of the searching window that is allowed was investigated by Marr-Poggio in [8] and Nasrabadi in [25]. As the centroids in vertical direction are not likely to be displaced, this istrue in the results, we use

24 instead of $15 \times W_\sigma$ as the vertical window size.

To obtain the actual pixel disparities between the pixels of matched curve-segments, we simply subtract the y-coordinates of the corresponding pixels, i.e., pixels that have the same y-coordinates. There are two cased where it is difficult to form a one-to-one pixel correspondence. The first is when part of the curve is occluded. The second is when several pixels on a curve have the same x-coordinates as in horizontal lines. In these cases, we set the pixel disparities equal to the curve's centroid disparity. This is an advantage of the technique compared to the local stereo matching techniques where horizontal lines are ignored.

Mismatch	Point Matching	Curve Matching	Current Tech
Tab	<20%	<0.0001%	<0.0001%
Jack		<1%	<0.01%
Screwdriver		<1%	<0.01%

Appendix A

Original Simulation Program Listing

A.1 Definitions DEF.H and DMASK.H

A.1.1 DEF.H

```
#define NDIM       2
#define DIME     512
#define RDIME    480
#define GREY     256
#define HEAD      64
#define MD         3
#define VARI      30.0
#define HGATE     64.0
#define LGATE     32.0
#define ANGLE     10.0*3.1415926535897932*255/180.0
#define ANGLES     6.0*3.1415926535897932*255/180.0
#define PI         3.1415926535897932
#define SHORT     32
#define MAX_SEG 1024
#define LENGTH  1024
#define DIPAT     32
#define DIPAJ     32
#define DIPAS     32
#define SIMS      10

typedef struct{
  float real;
  float imag;
}complex;

float image[DIME][DIME];
int   head[HEAD];
float img[DIME][DIME];
```

```
float grad[DIME][DIME];
float orie[DIME][DIME];
int   sl[MAX_SEG][3][LENGTH];
int   sm[MAX_SEG][3][LENGTH];
int   s[MAX_SEG][3][LENGTH];
int   segnl, segnm, segn;

float *vector();
int   *ivector();
float absl();
short sign();
float com();
```

A.1.2 DMASK.H

```
float m_x[MD][MD]={
  {-1,-2,-1},
  { 0, 0, 0},
  { 1, 2, 1}
};

float m_y[MD][MD]={
  {-1, 0, 1},
  {-2, 0, 2},
  {-1, 0, 1}
};

float m_l[MD][MD]={
  { 0,-1, 0},
  {-1, 4,-1},
  { 0,-1, 0}
};
```

A.2 Main Simulation Program

A.2.1 VZ0VIS.C

```
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include "def.h"

main(argc,argv)
int argc;
char *argv[];
{
  int *nn;
  complex *data, *data_tmp, *filter;
  float tmp[DIME][DIME];
```

```
  float mask[MD][MD];

  int i,j;

  FILE *fp, *fopen();

  /* read the original image file into */
  /* array 'image'.                    */
  fp = fopen(argv[1],"r");
  filerw(fp, 'r', image);

  /* initializing */
  nn     = ivector(0,NDIM);
  data   = (complex *) vector(0,DIME*DIME*2);
  data_tmp = (complex *) vector(0,DIME*DIME*2);
  filter = (complex *) vector(0,DIME*DIME*2);
  init(data,filter);
  nn[0] = DIME;
  nn[1] = DIME;

  /* FFT of image file and the Laplacian of Gaussian filter */
  /* 'img' is the profile of the convoled image             */
  fourn(data,nn,NDIM,1);
  fourn(filter,nn,NDIM,1);
  multiply(data,filter,data_tmp);
  fourn(data_tmp,nn,NDIM,-1);
  cmpl_int(data_tmp,img);

  /* convolution using the window of     */
  /* Laplacian of Gaussian filter        */
  /*initmask(mask);       */
  /*convl(mask,image,img);*/

  /* zero-crossing generator using 'img' */
  zero();
  fp = fopen(argv[2],"w");
  filerw(fp, 'w', img);
  printf("Zero-Crossing Done ...\n");

  /* gradient and direction of the edges */
  grad_orie();
  fp = fopen(argv[3],"w");
  filerw(fp, 'w', grad);
  fp = fopen(argv[4],"w");
  filerw(fp, 'w', orie);
  printf("Gradient Done ...\n");
}

filerw(fp, frw, imgp)
char frw;
float imgp[DIME][DIME];
```

```
FILE *fp;
{
  int i, j, m;

  if(frw=='r'){
    for(i=0; i<HEAD; ++i){
      head[i] = getc(fp);
    }

    for(i=0; i<DIME; ++i){
      for(j=0; j<DIME; ++j){
        if(i<RDIME){
          m = getc(fp);
          imgp[i][j] = (float) m;
        }else{
          imgp[i][j] = 0.0;
        }
      }
    }
  }else{
    for(i=0; i<HEAD; ++i){
      putc(head[i],fp);
    }
    for(i=0; i<RDIME; ++i){
      for(j=0; j<DIME; ++j){
        m = imgp[i][j] + 0.5;
        putc(m, fp);
      }
    }
  }
  fclose(fp);
}
```

A.2.2 VZ0VIS1.C

```
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include "def.h"

main(argc,argv)
int argc;
char *argv[];
{
  int i,j;

  FILE *fp, *fopen();

  fp = fopen(argv[1],"r");
  filerw(fp, 'r', grad);
```

```
  fp = fopen(argv[2],"r");
  filerw(fp, 'r', orie);
  tracking();
  fp = fopen(argv[3],"w");
  filerw(fp, 'w', img);
  hough();
  conver(sl,&segnl);

  fp = fopen(argv[4],"r");
  filerw(fp, 'r', grad);
  fp = fopen(argv[5],"r");
  filerw(fp, 'r', orie);
  tracking();
  fp = fopen(argv[6],"w");
  filerw(fp, 'w', img);
  hough();
  conver(sm,&segnm);

  fp = fopen(argv[7],"r");
  filerw(fp, 'r', grad);
  fp = fopen(argv[8],"r");
  filerw(fp, 'r', orie);
  tracking();
  fp = fopen(argv[9],"w");
  filerw(fp, 'w', img);
  hough();

  for(i=0; i<RDIME; ++i){
    for(j=0; j<DIME; ++j){
      image[i][j] = 0;
    }
  }

  sim(sl,segnl,sm,segnm);
  sim(sl,segnl,s,segn);
  sim(sm,segnm,s,segn);
  org();
  fp = fopen(argv[10],"w");
  filerw(fp, 'w', image);
}
```

A.3 Fast Fourier Transform VZ1FFT.C

```
#include <math.h>
#include "def.h"

init(data,filter)
complex *data, *filter;
{
  int i,j,idx;
```

```
  float m;

  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      idx = j+i*DIME;
      data[idx].real = image[i][j];
      data[idx].imag = 0;
      m=((float)((i-DIME/2.)*(i-DIME/2.)+(j-DIME/2.)
        *(j-DIME/2.)))/VARI;
      filter[idx].real = (m-2.0)*exp(-m/2.0);
      filter[idx].imag = 0;
    }
  }
}

int *ivector(n,nh)
int n,nh;
{
  int *v;

  v = (int *)malloc((unsigned)(nh-n+1)*sizeof(int));
  return v-n;
}

float *vector(n,nh)
int n,nh;
{
   float *v;

   v = (float *)malloc((unsigned)(nh-n+1)*sizeof(float));
   return v-n;
}

multiply(a,b,c)
complex *a, *b, *c;
{
  int i,x,y,z,ntot;
  float *data_tmp;

  for(y=0; y<DIME; ++y){
    for(x=0; x<DIME; ++x){
      z = y+x*DIME;
      c[z].real = (a[z].real*b[z].real-a[z].imag*b[z].imag)
                  *cos(PI*x)*cos(PI*y);
      c[z].imag = (a[z].real*b[z].imag+a[z].imag*b[z].real)
                  *cos(PI*x)*cos(PI*y);
    }
  }
  ntot = DIME*DIME;
  data_tmp = (float *) c;
  for(i=0; i<ntot*2; ++i){
```

```
    data_tmp[i] *= ntot;
  }
}

#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

fourn(data,nn,ndim,isign)
complex *data;
int nn[],ndim,isign;
{
  int i1,i2,i3,i2rev,i3rev,ip1,ip2,ip3,ifp1,ifp2;
  int ibit,idim,k1,k12,k2,k21,n,nprev,nrem,ntot;
  float tempi,tempr,*data_tmp;
  double theta,wi,wpi,wpr,wr,wtemp;

  ntot=1;
  for(idim=1;idim<=ndim;idim++)
    ntot *= nn[idim-1];

  nprev=1;
  for(idim=ndim;idim>=1;idim--){
    n=nn[idim-1];
    nrem=ntot/(n*nprev);
    ip1=nprev * 2;
    ip2=ip1*n;
    ip3=ip2*nrem;
    i2rev=1;

    for(i2=1;i2<=ip2;i2+=ip1){
      if(i2<i2rev){
        for(i1=i2;i1<=(i2+ip1-2);i1+=2){
          for(i3=i1;i3<=ip3;i3+=ip2){
            i3rev=i2rev+i3-i2;
            SWAP(data[i3/2].real,data[i3rev/2].real);
            SWAP(data[i3/2].imag,data[i3rev/2].imag);
          }
        }
      }
      ibit=ip2/2;
      while((ibit >= ip1) && (i2rev > ibit)){
        i2rev -= ibit;
        ibit = ibit/2;
      }
      i2rev += ibit;
    }

    ifp1=ip1;
    while(ifp1 < ip2){
      ifp2=ifp1 * 2;
      theta=((-isign) * 2 * M_PI)/(ifp2/ip1);
/* PI in last line was M_PI originally. */
```

```
      wtemp=sin(0.5*theta);
      wpr= -2.0*wtemp*wtemp;
      wpi=sin(theta);
      wr=1.0;
      wi=0.0;
      for(i3=1;i3<=ifp1;i3+=ip1){
        for(i1=i3;i1<=i3+ip1-2;i1+=2){
          for(i2=i1;i2<=ip3;i2+=ifp2) {
            k1=i2;
            k12=k1/2;
            k2=k1+ifp1;
            k21=k2/2;
            tempr=wr*(data[k21].real)-wi*(data[k21].imag);
            tempi=wr*(data[k21].imag)+wi*(data[k21].real);
            (data[k21].real)=(data[k12].real)-tempr;
            (data[k21].imag)=(data[k12].imag)-tempi;
            (data[k12].real) += tempr;
            (data[k12].imag) += tempi;
          }
        }
        wtemp=wr;
        wr =wr*wpr-wi*wpi+wr;
        wi=wi*wpr+wtemp*wpi+wi;
      }
      ifp1=ifp2;
    }
    nprev *= n;
  }
  if(isign==1){
    data_tmp =(float *)data;
    for(i1=0;i1<(ntot*2);++i1)  data_tmp[i1] /= ntot;
  }
}
```

A.4 Function Group VZ2SUB.C

```
#include <math.h>
#include "def.h"
#include "dmask.h"

conver(st,seg)
int st[MAX_SEG][3][LENGTH];
int *seg;
{
  int i,j,k;

  *seg = segn;
  for(i=0; i<segn; ++i){
    st[i][0][0] = s[i][0][0];
    st[i][1][0] = s[i][1][0];
```

```
    st[i][2][0] = s[i][2][0];
    for(j=1; j<=st[i][2][0]; ++j)
      for(k=0; k<3; ++k)
        st[i][k][j] = s[i][k][j];
  }
}

convl(mask,img_i,img_o)
float mask[MD][MD];
float img_i[DIME][DIME],
      img_o[DIME][DIME];
{
  int i, j, ipn, ipp, jpn, jpp;
  int mi, mj;
  int is, js;
  float sum;

  for(i=0; i<DIME; ++i){
    defl(i,&ipn,&mi);
    defh(i,&ipp);

    for(j=0; j<DIME; ++j){
      defl(j,&jpn,&mj);
      defh(j,&jpp);
      sum=0.0;
      for(is=ipn; is<=ipp; ++is){
        for(js=jpn; js<=jpp; ++js){
          sum += mask[is-ipn+mi][js-jpn+mj]*img_i[is][js];
        }
      }
      img_o[i][j]=sum;
    }
  }
}

pconvl(mask,i,j,tmp)
float mask[MD][MD];
float *tmp;
int *i,*j;
{
  int ipn, ipp, jpn, jpp;
  int mi, mj;
  int is, js;
  float sum;

  defl(*i,&ipn,&mi);
  defh(*i,&ipp);
  defl(*j,&jpn,&mj);
  defh(*j,&jpp);
  sum=0.0;
  for(is=ipn; is<=ipp; ++is){
```

```
    for(js=jpn; js<=jpp; ++js){
      sum += mask[is-ipn+mi][js-jpn+mj]*image[is][js];
    }
  }
  *tmp=sum;
}

cmpl_int(data,tmp)
complex *data;
float tmp[DIME][DIME];
{
  int i, j, idx;

  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      idx = j+i*DIME;
      tmp[i][j] = data[idx].real;
    }
  }
}

zero()
{
  int i,j;
  float tmp[DIME][DIME];

  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      tmp[i][j] = 0;
    }
  }

  for(i=0; i<DIME; ++i){
    for(j=0; j<(DIME-2); ++j){
      if(img[i][j] == 0) continue;
      if(sign(img[i][j],img[i][j+1],img[i][j+2]) == 1){
        tmp[i][j] = 255;
      }
    }
  }

  for(j=0; j<DIME; ++j){
    for(i=0; i<(DIME-2); ++i){
      if(img[i][j] == 0) continue;
      if(sign(img[i][j],img[i+1][j],img[i+2][j]) == 1){
        tmp[i][j] = 255;
      }
    }
  }

  for(i=0; i<DIME; ++i){
```

```
    for(j=0; j<DIME; ++j){
      image[i][j] = img[i][j];
      img[i][j] = tmp[i][j];
    }
  }
}

short sign(x,y,z)
float x,y,z;
{
  if(x>0 && y<0){
    return 1;
  }else if(x<0 && y>0){
    return 1;
  }else if(x>0 && y==0 && z<0){
    return 1;
  }else if(x<0 && y==0 && z>0){
    return 1;
  }else{
    return 0;
  }
}

grad_orie()
{
  int i,j;
  double g,o,m,mm,n,nn,tmp;
  float maxo,maxg,tt;

  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      grad[i][j]=0.0;
      orie[i][j]=0.0;
    }
  }

  mm = 0.0001;
  nn = 300000.0;
  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      if(img[i][j] == 0) continue;
      pconvl(m_x,&i,&j,&tt);
      grad[i][j]=tt;
      pconvl(m_y,&i,&j,&tt);
      orie[i][j]=tt;
      m = grad[i][j];
      if(m>=0 && m<mm) m = mm;
      if(m<0 && m>-mm) m = -mm;
      n = orie[i][j];
      if(absl(n/m)>nn) tmp=nn;
      else tmp = n/m;
```

```
      o = atan(tmp)+PI/2.0;
      g = sqrt(m*m+n*n);
      grad[i][j] = g;
      orie[i][j] = o;
    }
  }
  maxo = 0.0;
  maxg = 0.0;
  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      if(orie[i][j]>maxo) maxo=orie[i][j];
      if(grad[i][j]>maxg) maxg=grad[i][j];
    }
  }
  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      orie[i][j]=255.0*orie[i][j]/maxo;
      grad[i][j]=255.0*grad[i][j]/maxg;
    }
  }
}

defl(i,l,m)
int i,*l,*m;
{
  if((i-1)<0){
    *l=0;
    *m=1-i;
  }else{
    *l=i-1;
    *m=0;
  }
}

defh(i,h)
int i,*h;
{
  if((i+1)>=DIME){
    *h=DIME-1;
  }else{
    *h=i+1;
  }
}

float absl(x)
float x;
{
  if(x>0){
    return x;
  }else{
    return -x;
```

```
  }
}

gray(tmp)
float tmp[DIME][DIME];
{
  int i,j;
  float max,min;

  max=64;
  min=0;
/*
  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      if(tmp[i][j]>max) max = tmp[i][j];
      if(tmp[i][j]<min) min = tmp[i][j];
    }
  }
*/
  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      tmp[i][j] = 255.0*(tmp[i][j]-min)/(max-min);
      if(tmp[i][j]>255) tmp[i][j]=255;
    }
  }
}

snr(orig,proc)
float orig[DIME][DIME];
float proc[DIME][DIME];
{
  int   i,j;
  float snr,m;

  snr=0.0;

  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      m = orig[i][j]-proc[i][j];
      snr+=m*m;
    }
  }
  snr = log10(((GREY-1)*(GREY-1)/snr)*DIME*DIME);
  printf("SNR = %8.3f(dB)\n",snr);
}

initmask(mask)
float mask[MD][MD];
{
  float m;
  int i,j;
```

```
  for(i=-MD/2; i<=MD/2; ++i){
    for(j=-MD/2; j<=MD/2; ++j){
      m = ((float) (i*i+j*j)) / VARI;
      mask[i+MD/2][j+MD/2] = (m-2.0)*exp(-m/2.0);
    }
  }
}
```

A.5 Curve Tracking VZ3SEG.C

```
#include "def.h"

tracking()
{
  int i,j,idx;
  int *p,*cx,*cy;
  int ii;

  p  = (int *) malloc(sizeof(int));
  cx = (int *) malloc(sizeof(int));
  cy = (int *) malloc(sizeof(int));

  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      img[i][j] = 0;
    }
  }

  idx = 0;
  for(i=0; i<DIME; ++i){
    for(j=0; j<DIME; ++j){
      *p  = 1;
      *cx = 0;
      *cy = 0;
      if(grad[i][j]<HGATE) continue;
      s[idx][0][*p] = i;
      s[idx][1][*p] = j;
      ++*p;
      move(i,j,idx,p,cx,cy);
      if(*p> SHORT){
        s[idx][0][0] = (int) (*cx/(*p)+0.5);
        s[idx][1][0] = (int) (*cy/(*p)+0.5);
        s[idx][2][0] = *p;
           img[s[idx][0][0]][s[idx][1][0]] = 255;
           img[s[idx][0][0]+1][s[idx][1][0]-1] = 255;
           img[s[idx][0][0]-1][s[idx][1][0]+1] = 255;
           img[s[idx][0][0]+1][s[idx][1][0]+1] = 255;
           img[s[idx][0][0]-1][s[idx][1][0]-1] = 255;
           for(ii=1;ii<=*p;++ii){
```

```
            img[s[idx][0][ii]][s[idx][1][ii]] = 255;
          }
        ++idx;
        if(idx-1 > MAX_SEG){
          printf("Too many segments!\n");
          exit(0);
        }
      }
    }
  }
  segn = idx;
}

move(i,j,idx,p,cx,cy)
int i,j,idx;
int *p,*cx,*cy;
{
  int is,js;
  int ish,jsl,jsh;
  int m;

  defl(j,&jsl,&m);
  defh(i,&ish);
  defh(j,&jsh);
  for(is=i; is<=ish; ++is){
    for(js=jsl; js<=jsh; ++js){
      if(is==i && js<=j) continue;
      if(grad[is][js]<LGATE ||
         absl(orie[is][js]-orie[i][j])>ANGLE)
        continue;
      s[idx][0][*p] = is;
      *cx += is;
      s[idx][1][*p] = js;
      *cy += js;
      grad[is][js] =0;  /* IMPORTANT */
      ++*p;
      if(*p > LENGTH){
        printf("Too many points in the segment!\n");
        exit(0);
      }
      move(is,js,idx,p,cx,cy);
      break;
    }
  }
}
```

A.6 Generalized Hough Transform VZ4HOU.C

```
#include "def.h"
```

```
hough()
{
  int cx,cy;
  int idx,p;
  int len;

  for(idx=0; idx<segn; ++idx){
    cx = s[idx][0][0];
    cy = s[idx][1][0];
    len = s[idx][2][0];
    for(p=1; p<=len; ++p){
      s[idx][2][p] = orie[s[idx][0][p]][s[idx][1][p]];
      s[idx][0][p] -= cx;
      s[idx][1][p] -= cy;
    }
  }
}
```

A.7 Matching Process VZ5SIM.C

```
#include "def.h"
#include <stdio.h>

sim(sma,sem,ss,ses)
int sma[MAX_SEG][3][LENGTH], sem;
int ss[MAX_SEG][3][LENGTH], ses;
{
  int hbuf;
  int cenxm, cenym, cenxs, cenys;
  int idxm, idxs, lenm, lens;
  int lmp, lsp;
  int i,j;
  int tmpx, tmpy;

  for(idxm=0; idxm<sem-1; ++idxm){
    cenxm = sma[idxm][0][0];
    cenym = sma[idxm][1][0];
    lenm  = sma[idxm][2][0];
    for(idxs=0; idxs<ses-1; ++idxs){
      hbuf = 0;
      cenxs = ss[idxs][0][0];
      cenys = ss[idxs][1][0];
      lens  = ss[idxs][2][0];
      if(absl(cenxm-cenxs)>15*sqrt(VARI) ||
         absl(cenym+DIPAT-cenys)>15*sqrt(VARI)) continue;
      for(lmp=1; lmp<=lenm; ++lmp){
        for(lsp=1; lsp<=lens; ++lsp){
          if(absl(sma[idxm][2][lmp]-ss[idxs][2][lsp])>ANGLES)
            continue;
            tmpx = cenxm+sma[idxm][0][lmp]-ss[idxs][0][lsp];
```

```
            tmpy = cenym+sma[idxm][1][lmp]-ss[idxs][1][lsp];
            if(absl(tmpx-cenxm)<1 && absl(tmpy-cenym)<1)
            ++hbuf;
        }
      }
      if(hbuf/lenm>SIMS){
        for(lmp=1; lmp<=lenm; ++lmp)
          image[cenxm+sma[idxm][0][lmp]]
               [cenym+sma[idxm][1][lmp]] = 255;
        for(lsp =1; lsp<=lens; ++lsp)
          image[cenxs+ss[idxs][0][lsp]]
               [cenys+ss[idxs][1][lsp]] = 255;
      }
    }
  }
}
```

Appendix B

Simulation Results

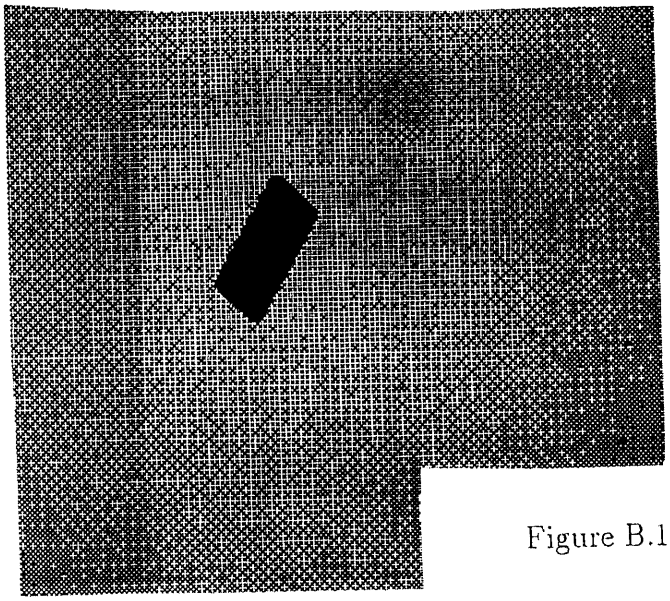
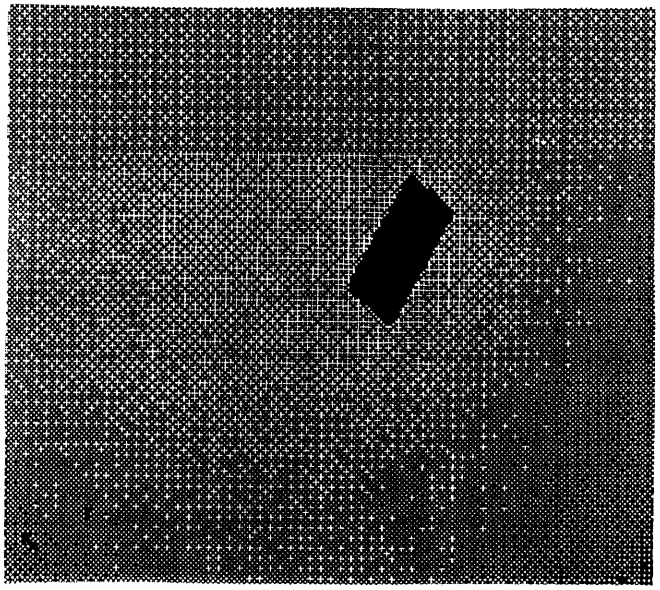
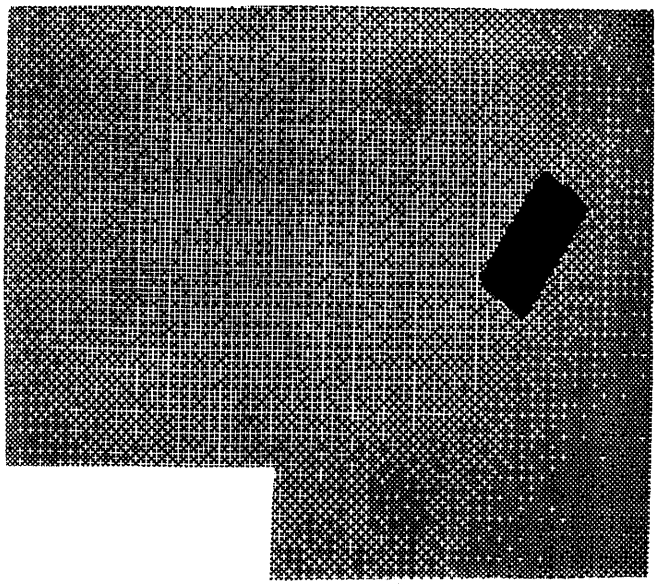


Figure B.1: Original Image of Tab





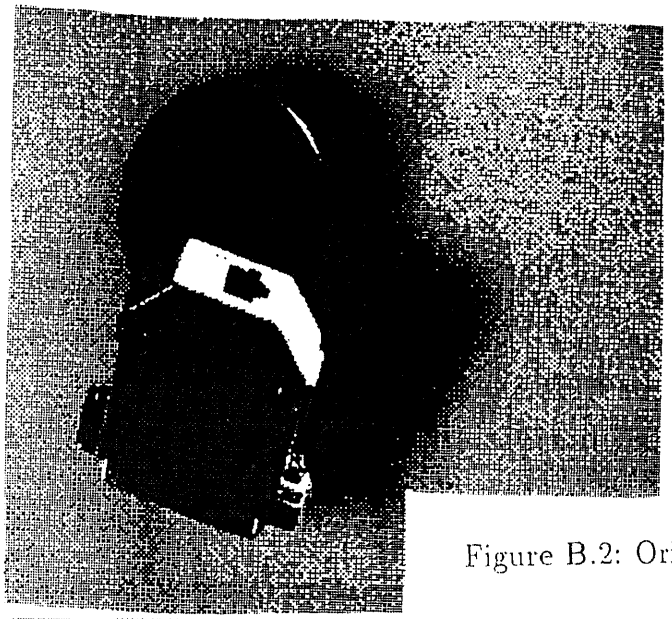
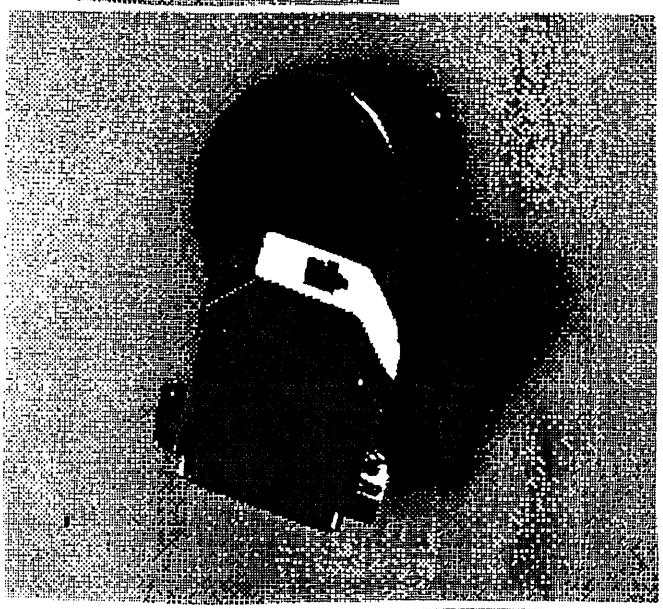
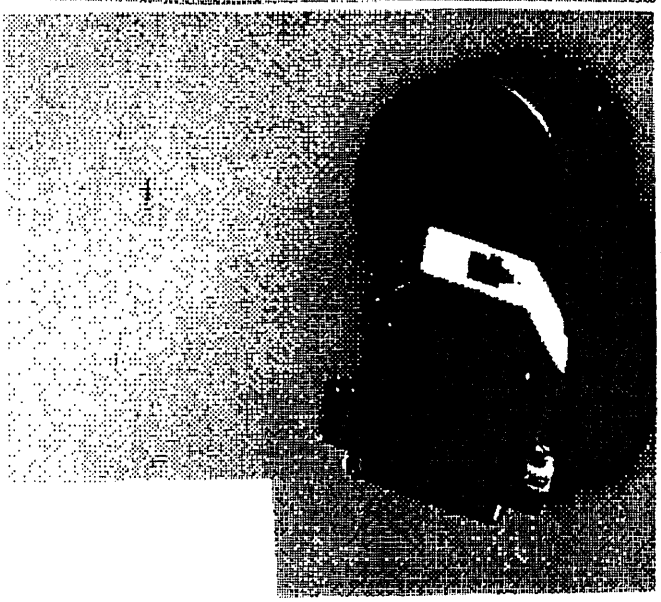


Figure B.2: Original Image of Jack





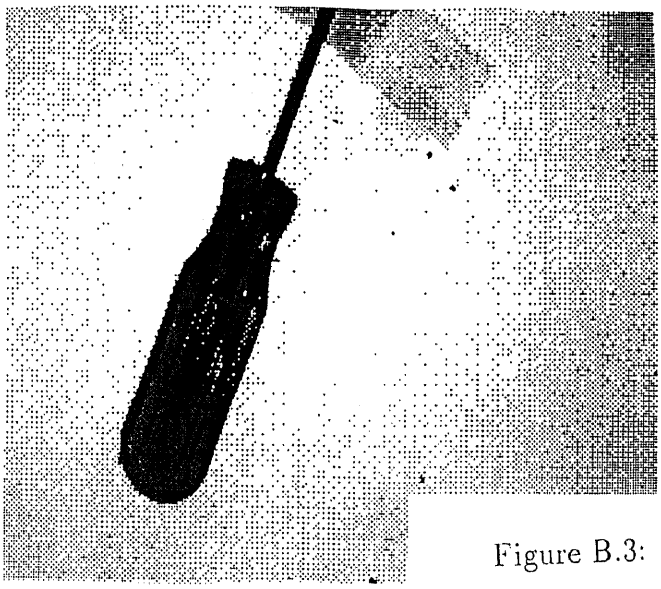
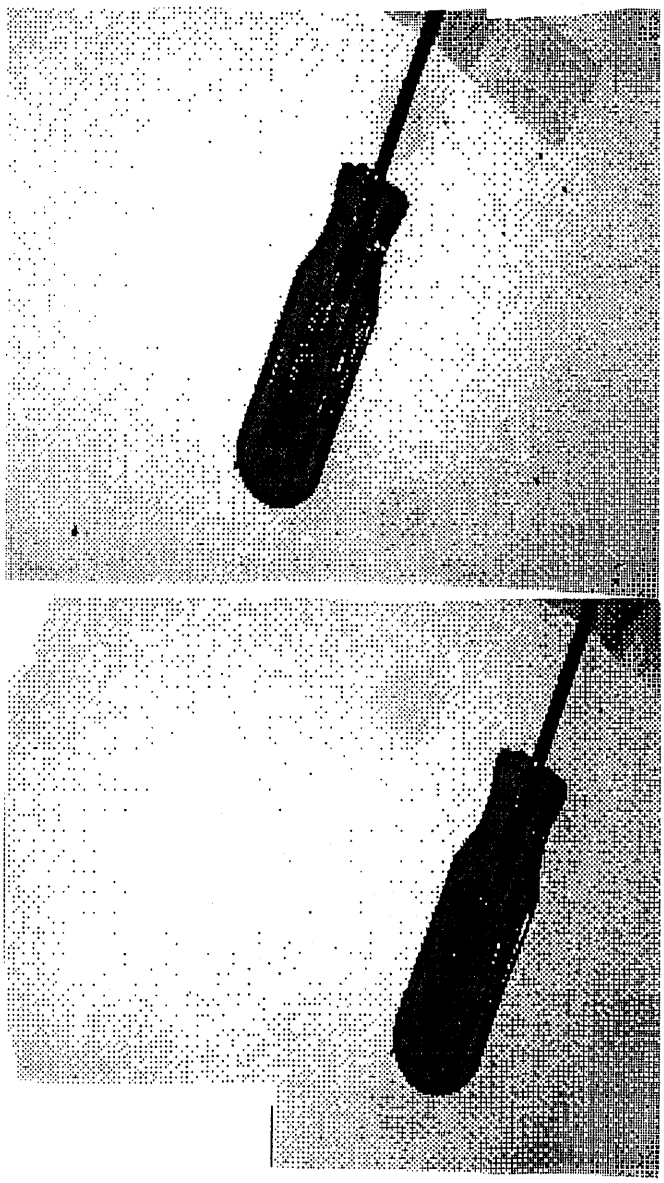


Figure B.3: Original Image of Screwdriver



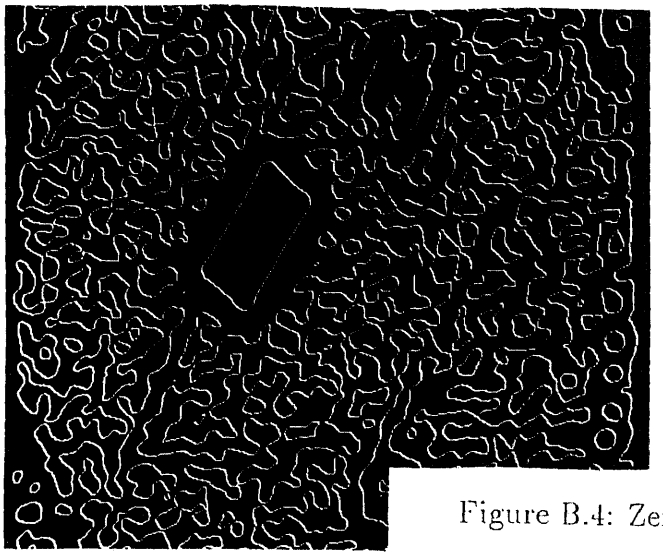
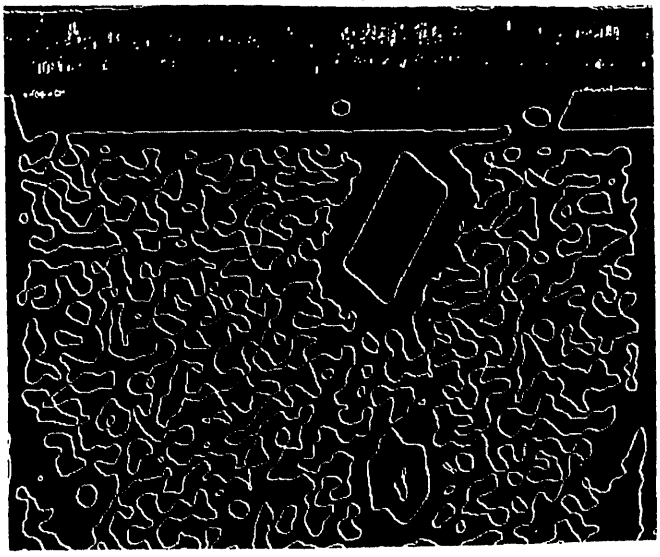
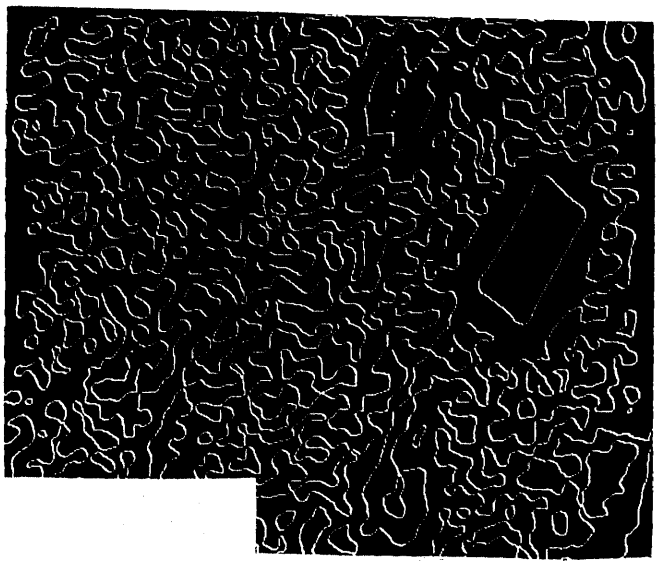


Figure B.4: Zero-Crossings of Tab at $\sigma^2 = 30$





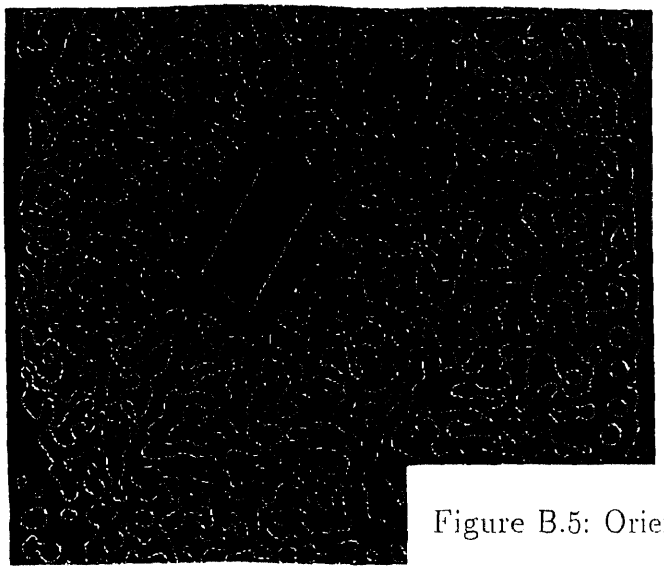
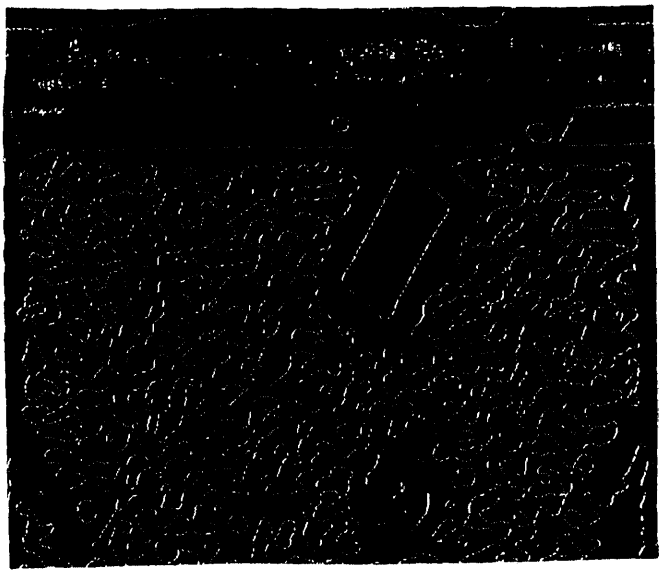
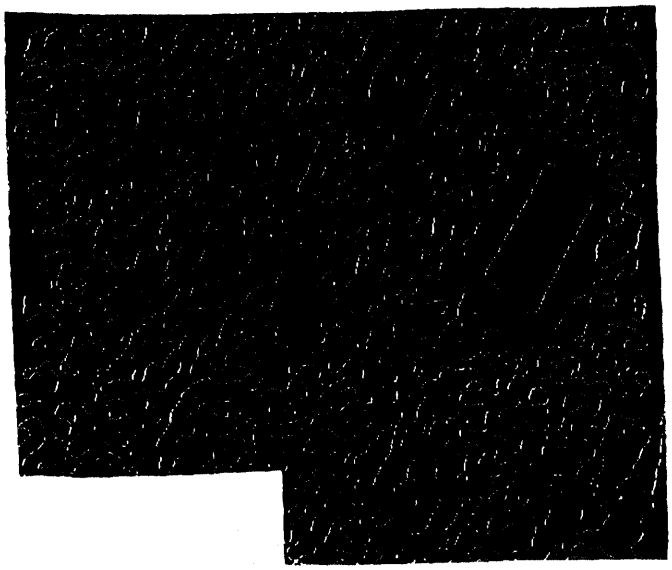


Figure B.5: Orientation of Tab Zero-Crossings at $\sigma^2 = 30$





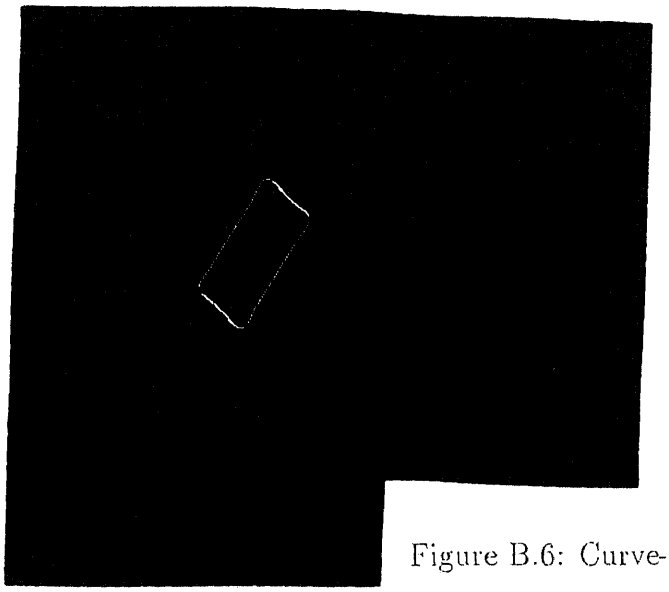


Figure B.6: Curve-Segmnets of Tab at $\sigma^2 = 30$

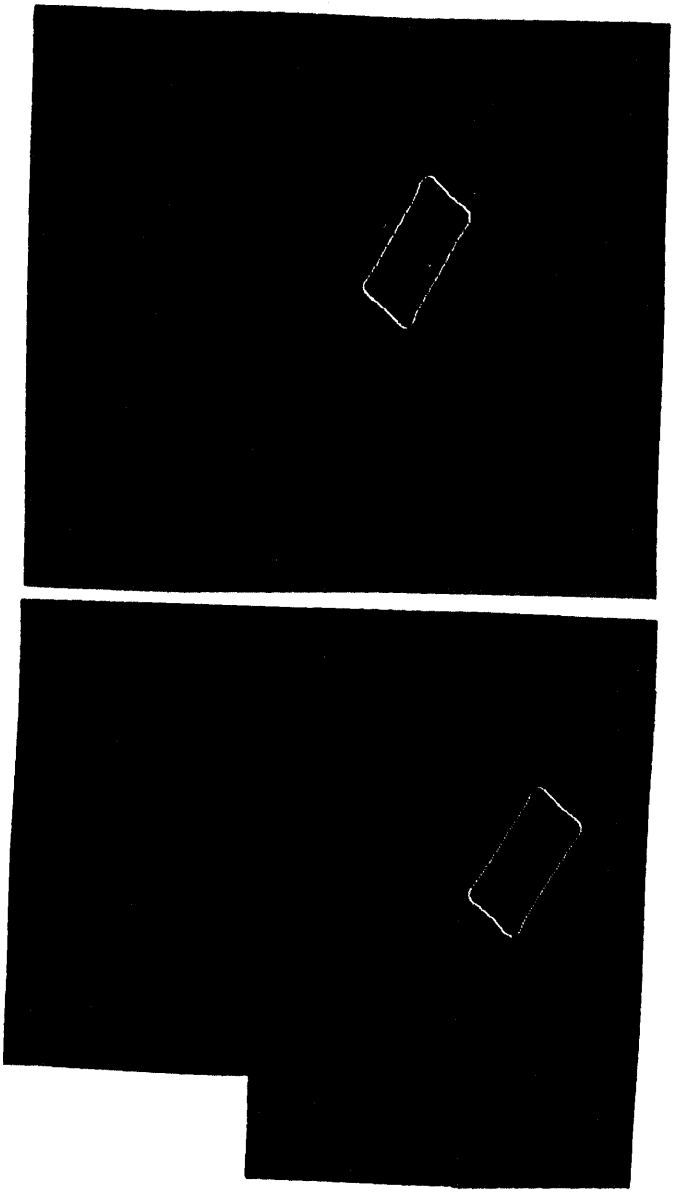
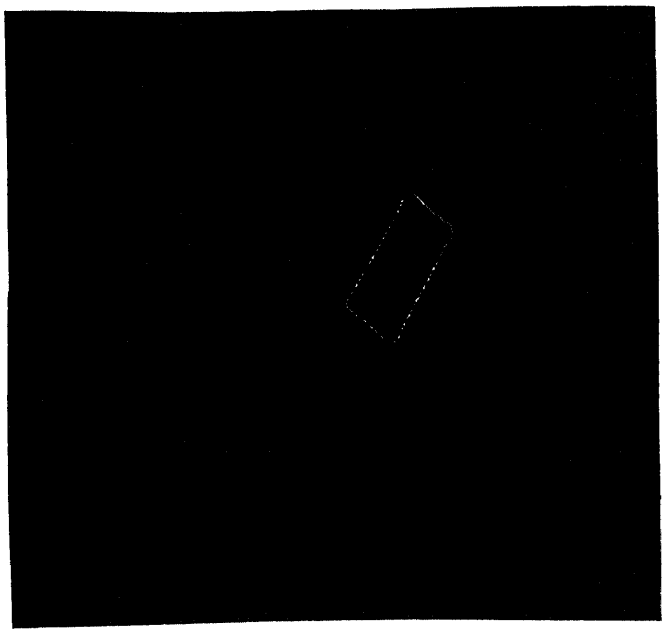


Figure B.7: Disparity Picture of Tab at $\sigma^2 = 30$



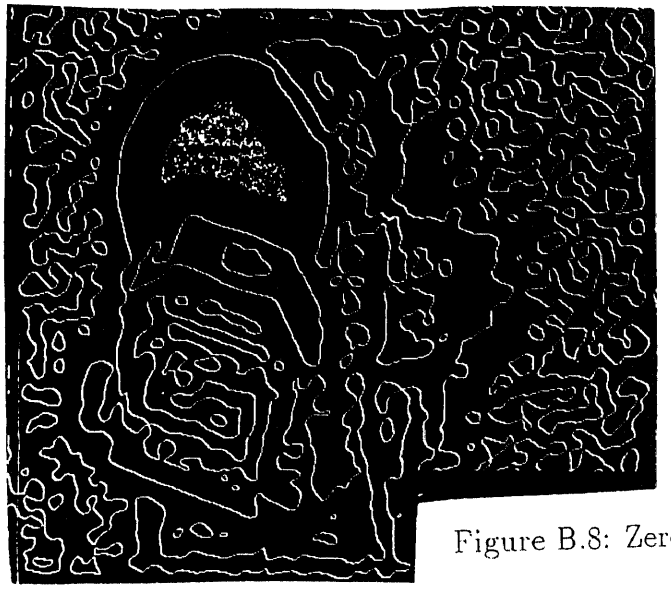
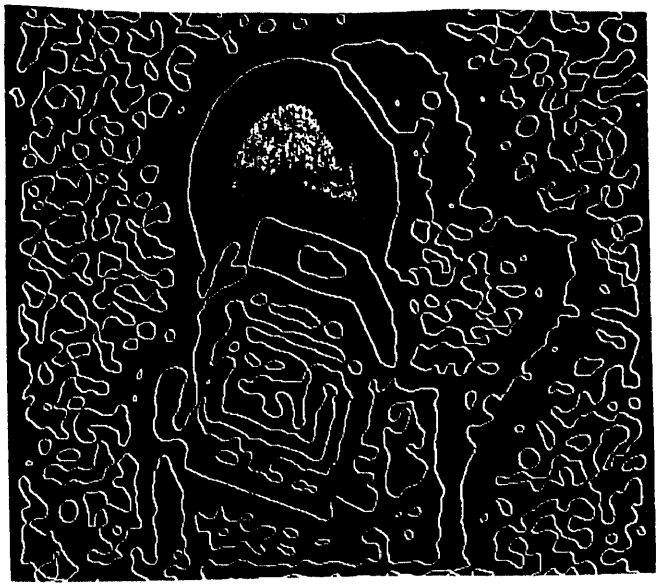
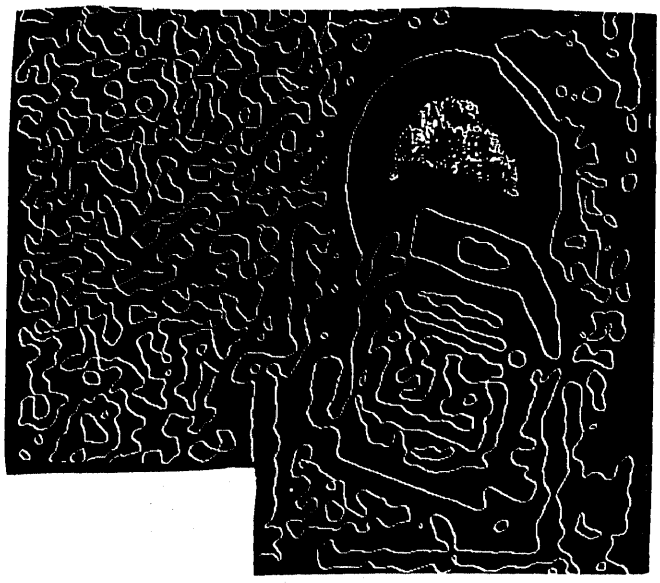


Figure B.8: Zero-Crossings of Jack at $\sigma^2 = 30$





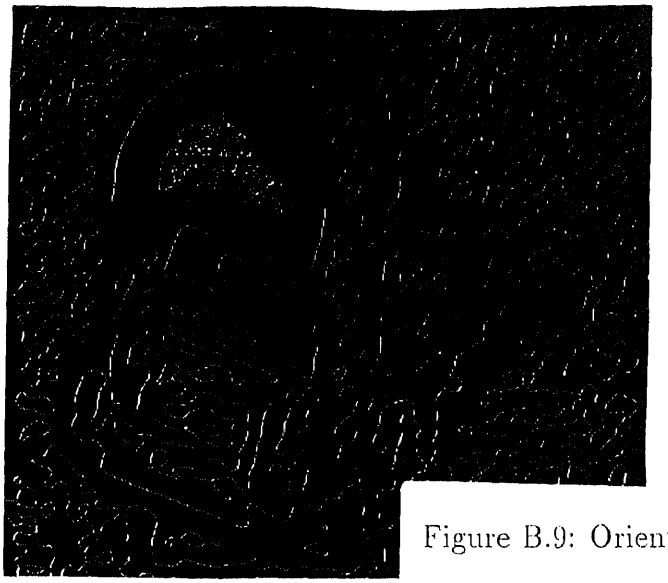
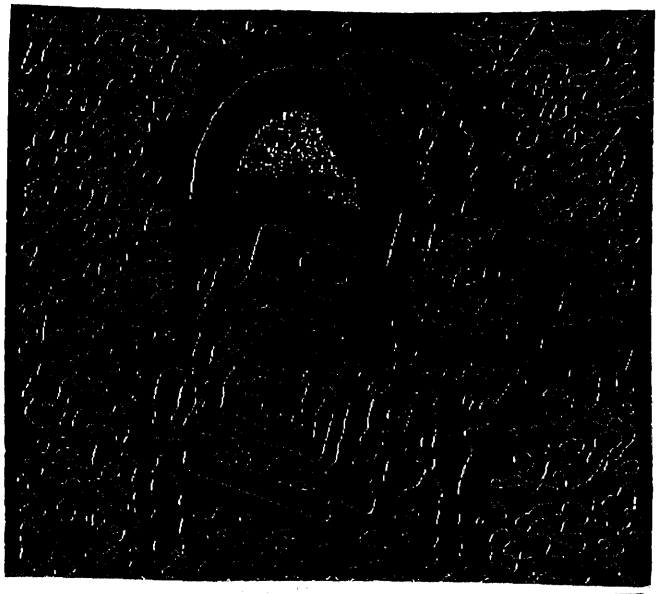
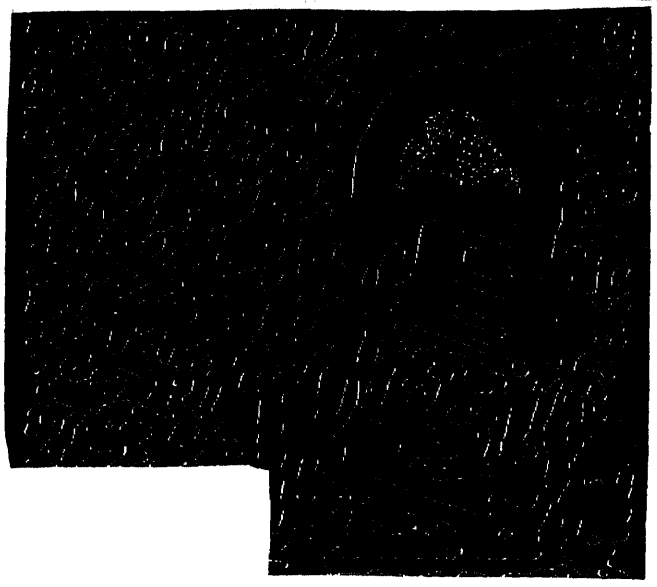


Figure B.9: Orientation of Jack Zero-Crossings at $\sigma^2 = 30$





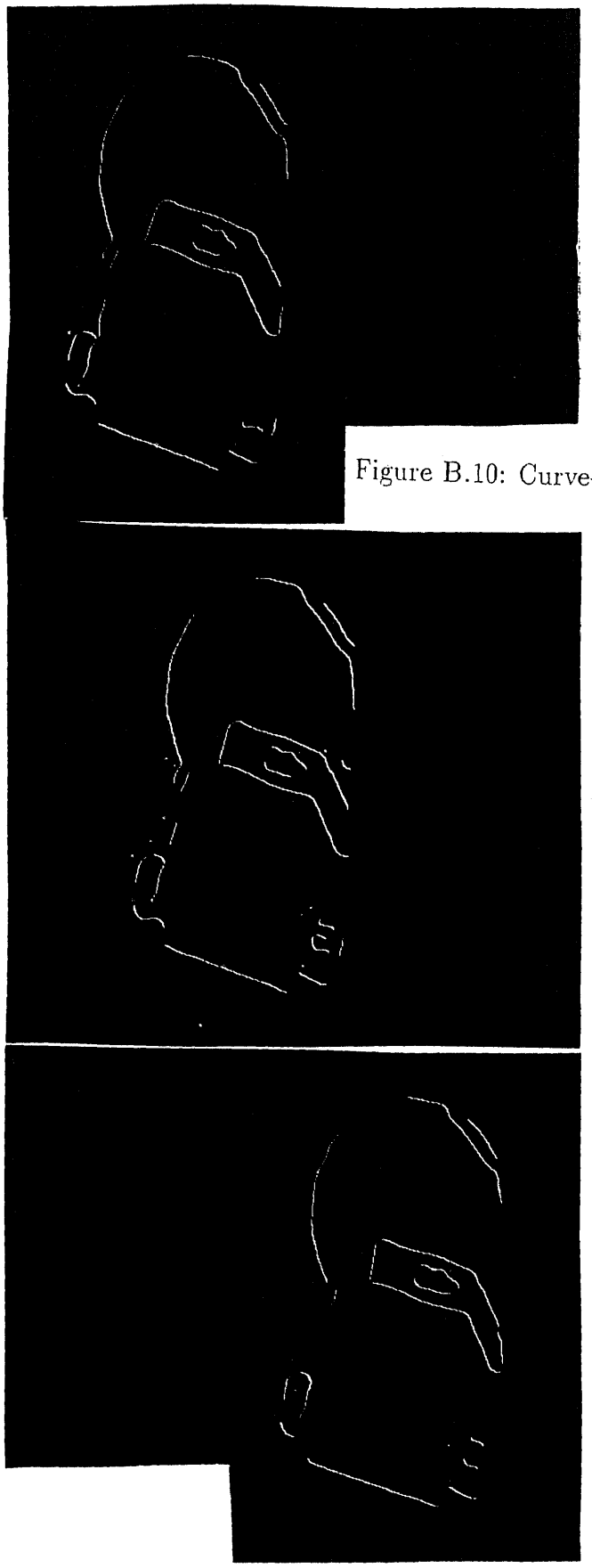
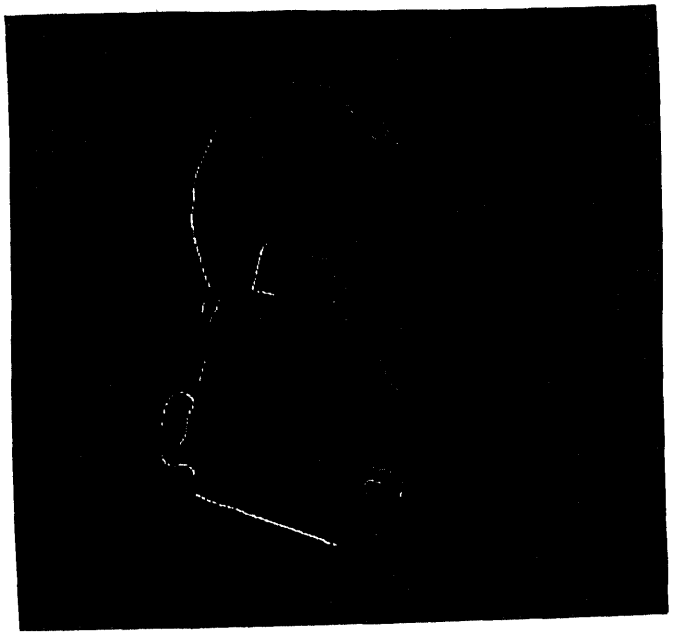


Figure B.10: Curve-Segmnets of Jack at $\sigma^2 = 30$

Figure B.11: Disparity Picture of Jack at $\sigma^2 = 30$



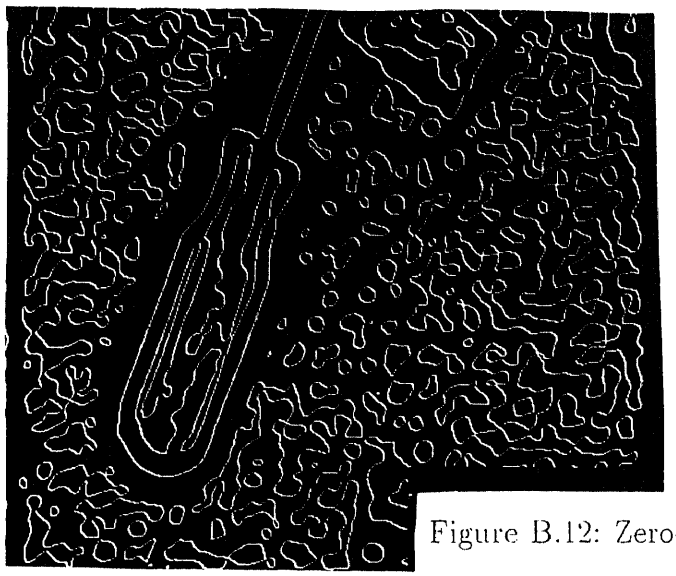
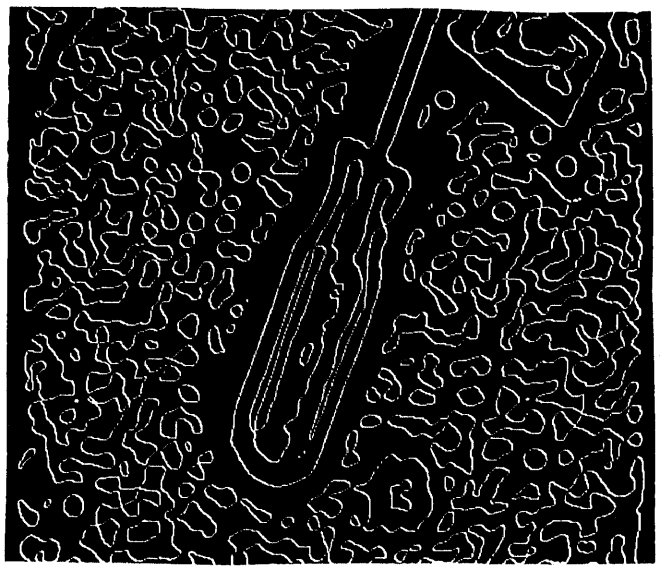
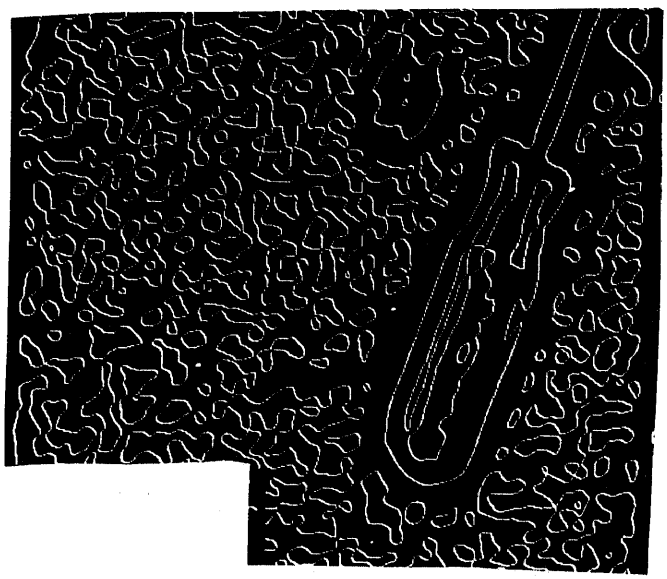


Figure B.12: Zero-Crossings of Screwdriver at $\sigma^2 = 30$





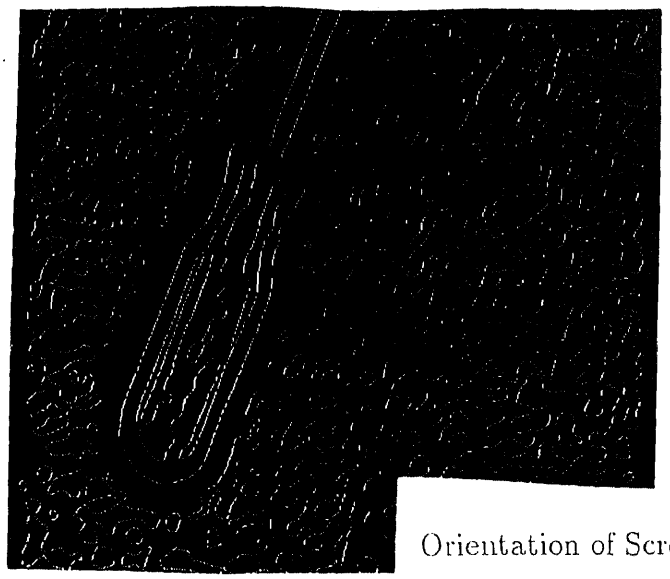
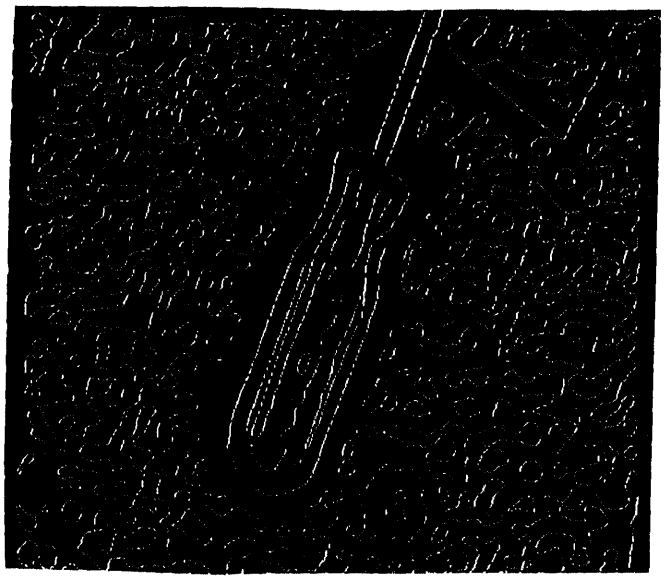
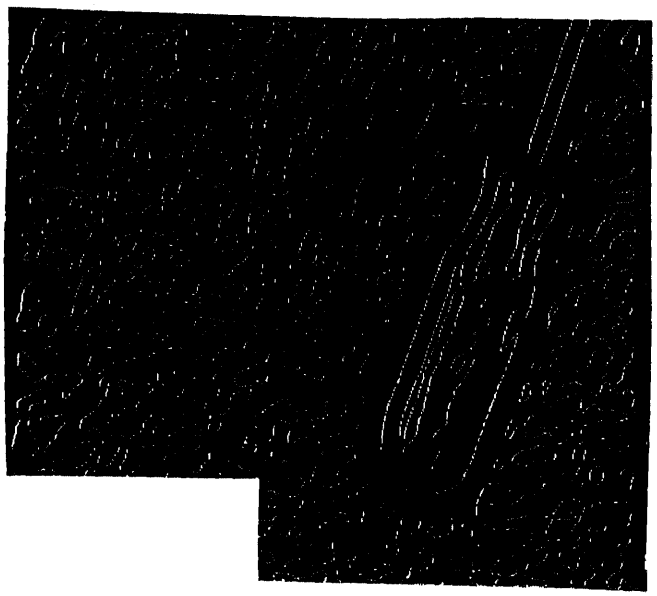


Figure B.13:

Orientation of Screwdriver Zero-Crossings at $\sigma^2 = 30$





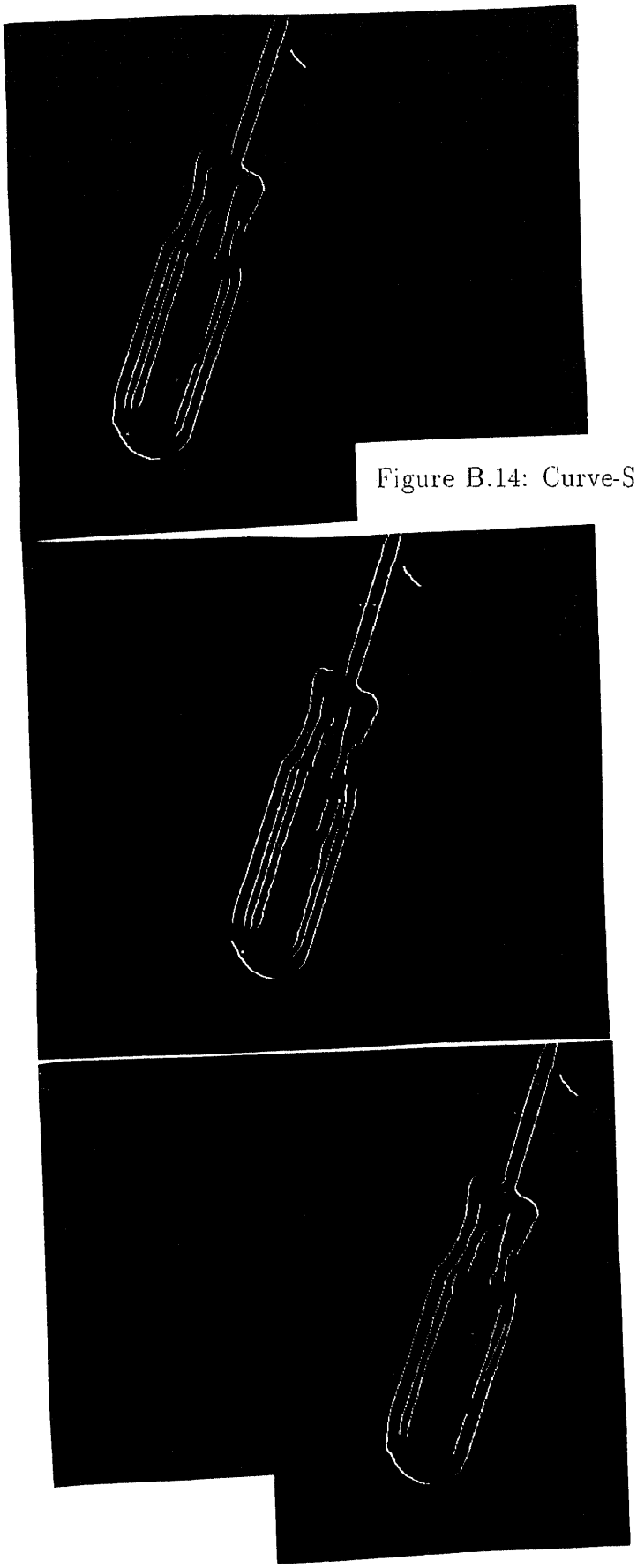
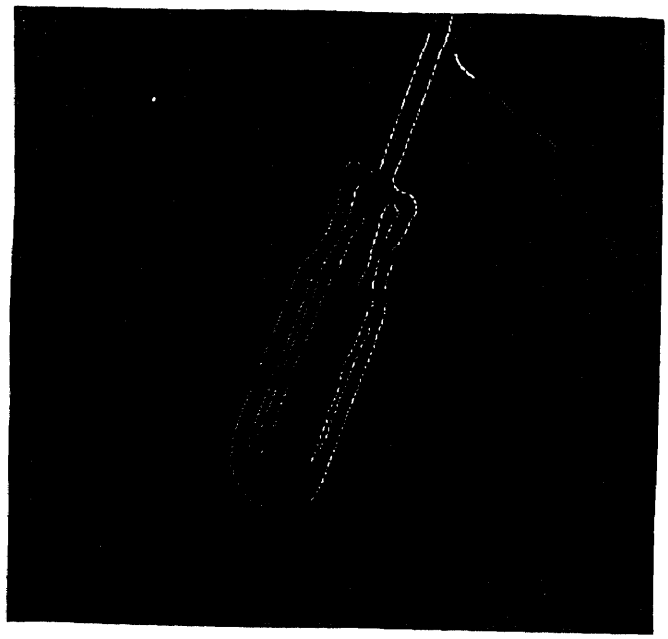


Figure B.14: Curve-Segmnets of Screwdriver at $\sigma^2 = 30$

Figure B.15: Disparity Picture of Screwdriver at $\sigma^2 = 30$



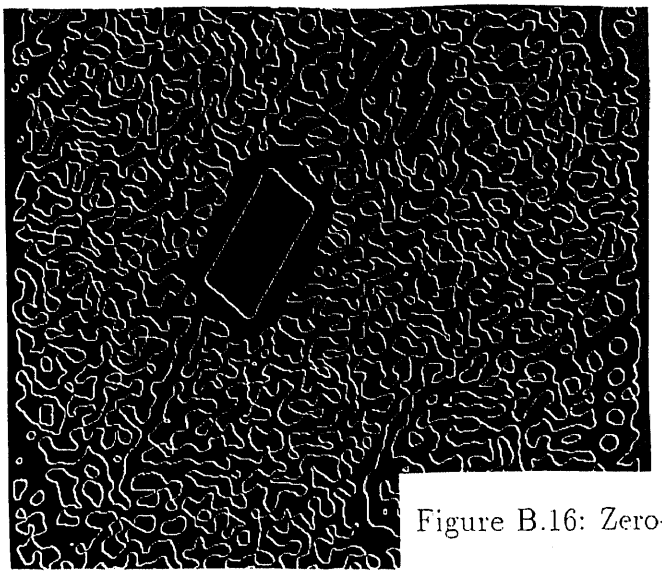
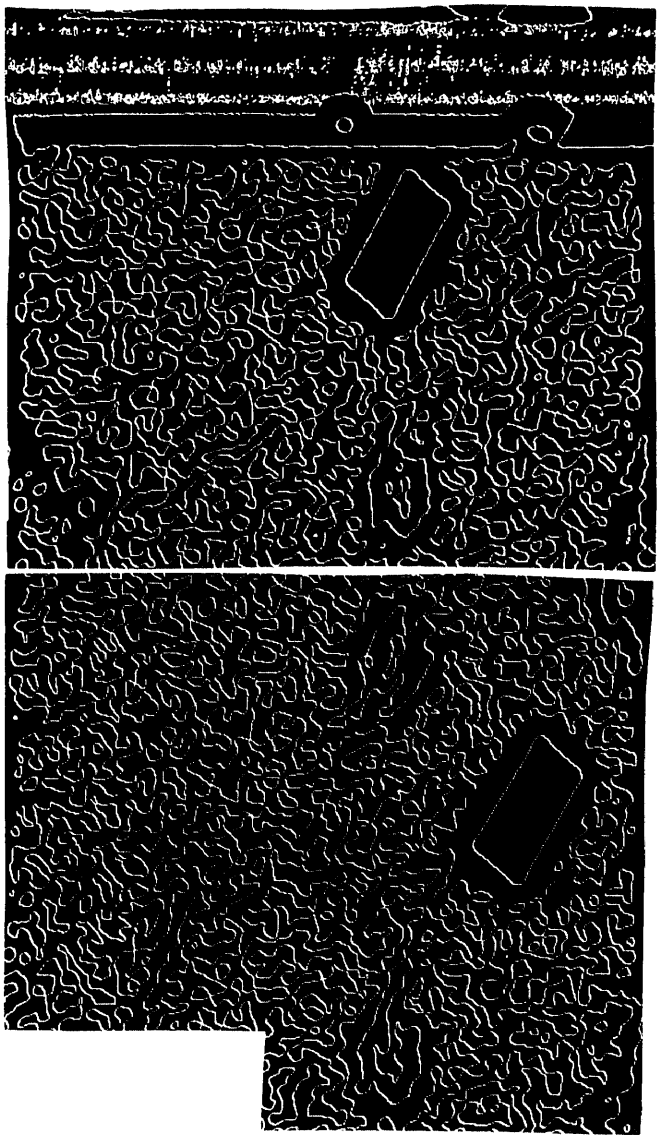


Figure B.16: Zero-Crossings of Tab at $\sigma^2 = 20$



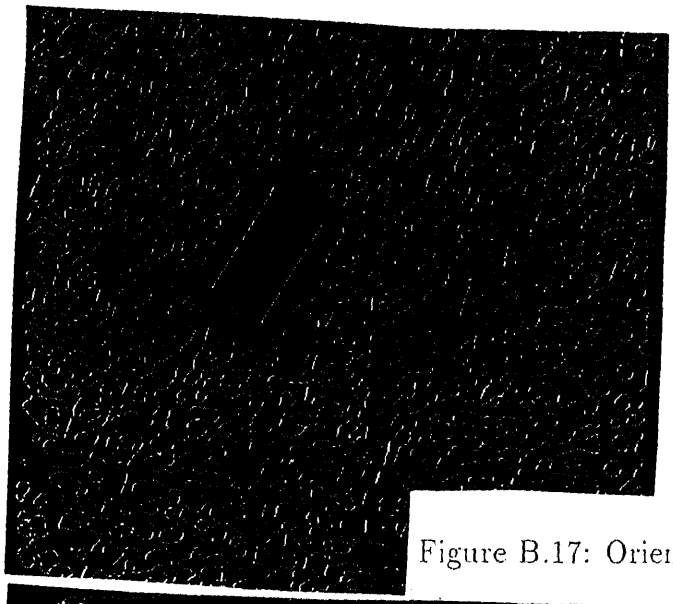
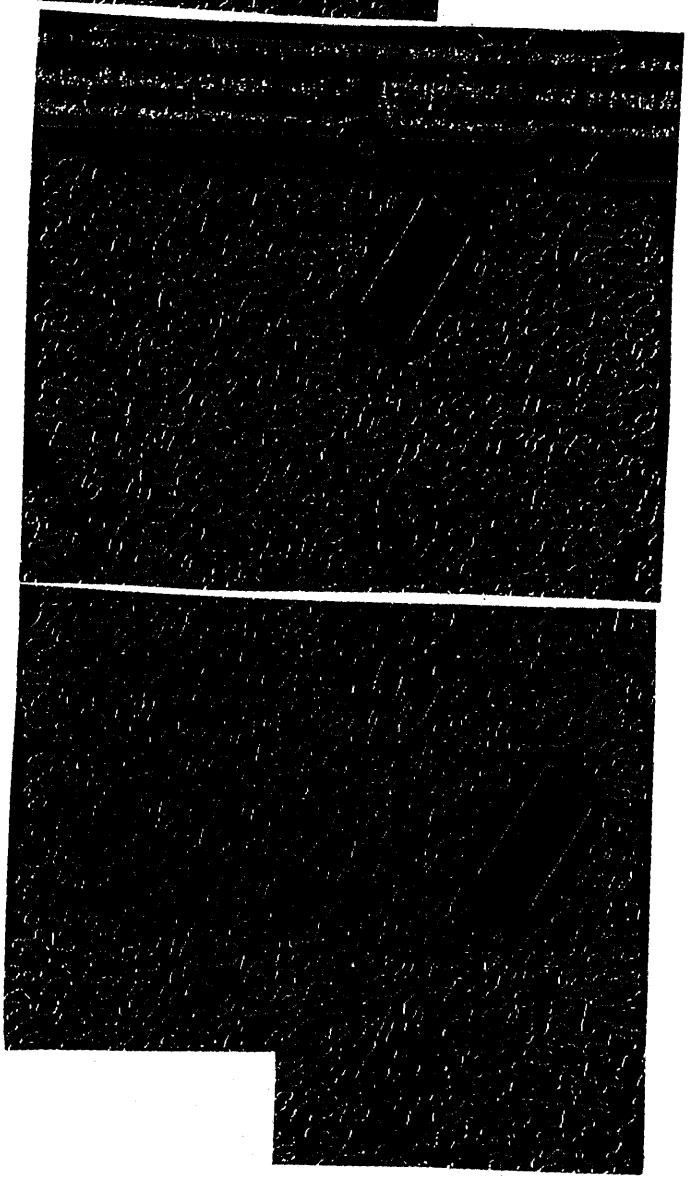


Figure B.17: Orientation of Tab Zero-Crossings at $\sigma^2 = 20$



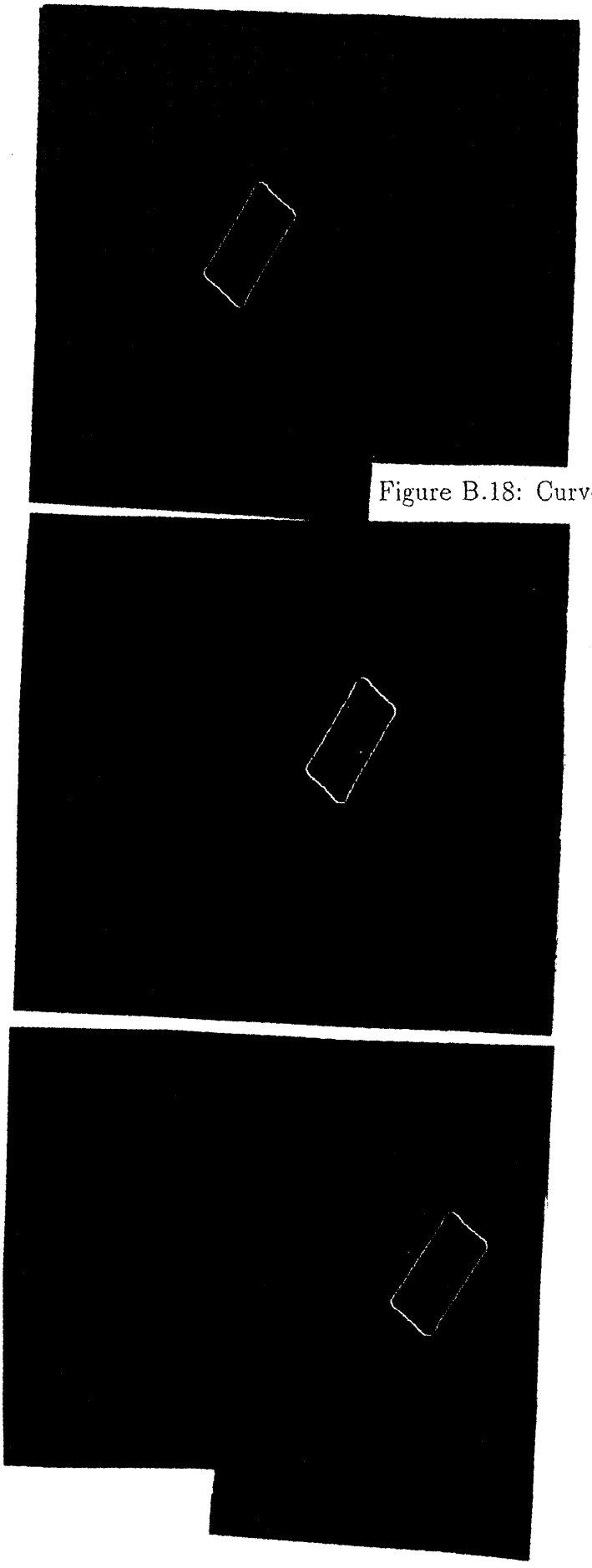
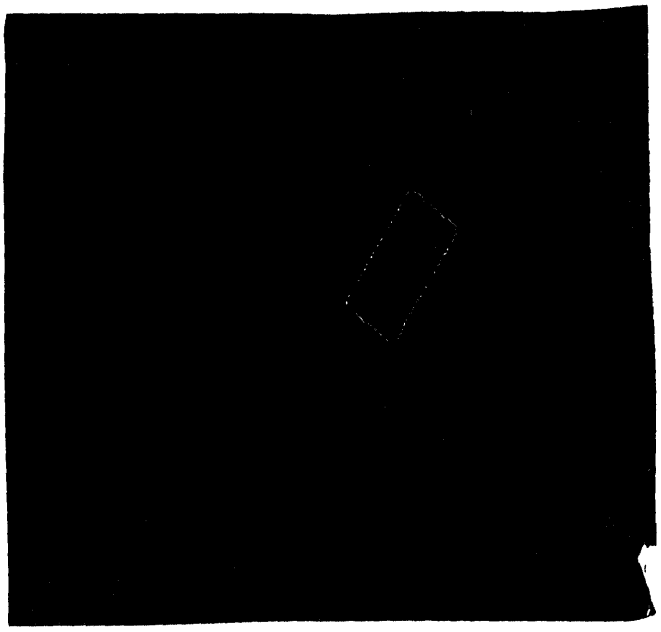


Figure B.18: Curve-Segmnets of '

Figure B.19: Disparity Picture of Tab at $\sigma^2 = 20$



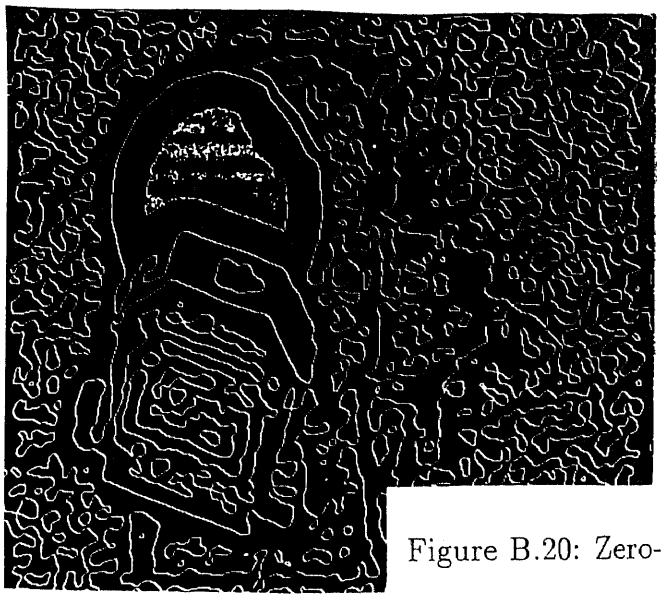
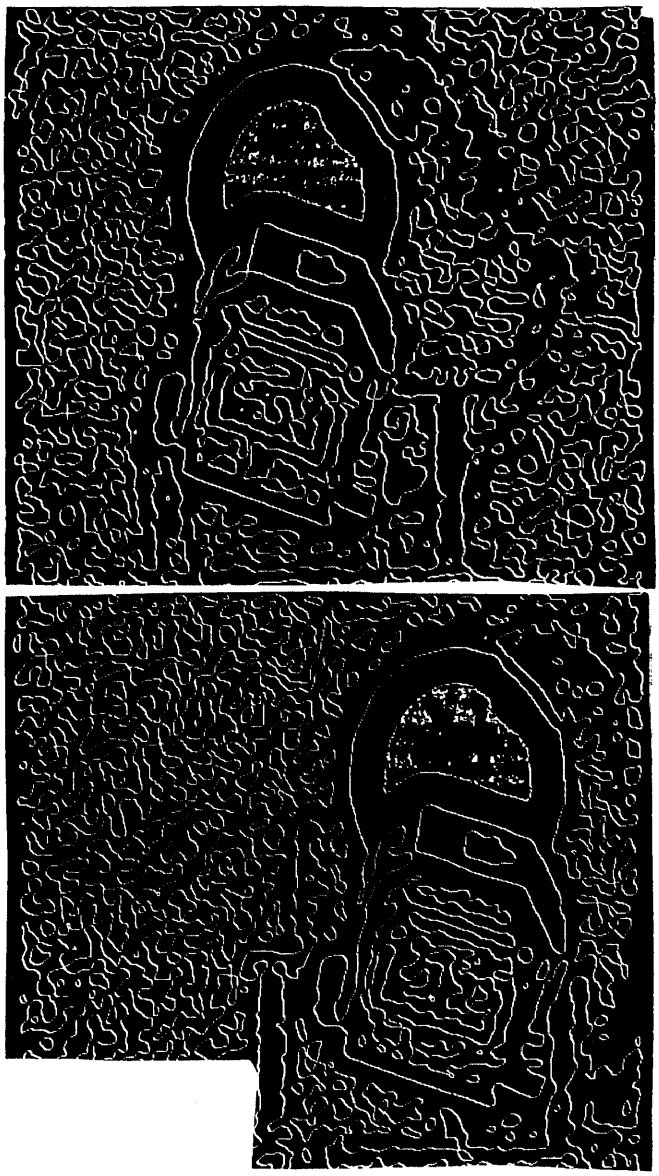


Figure B.20: Zero-Crossings of Jack at $\sigma^2 = 20$



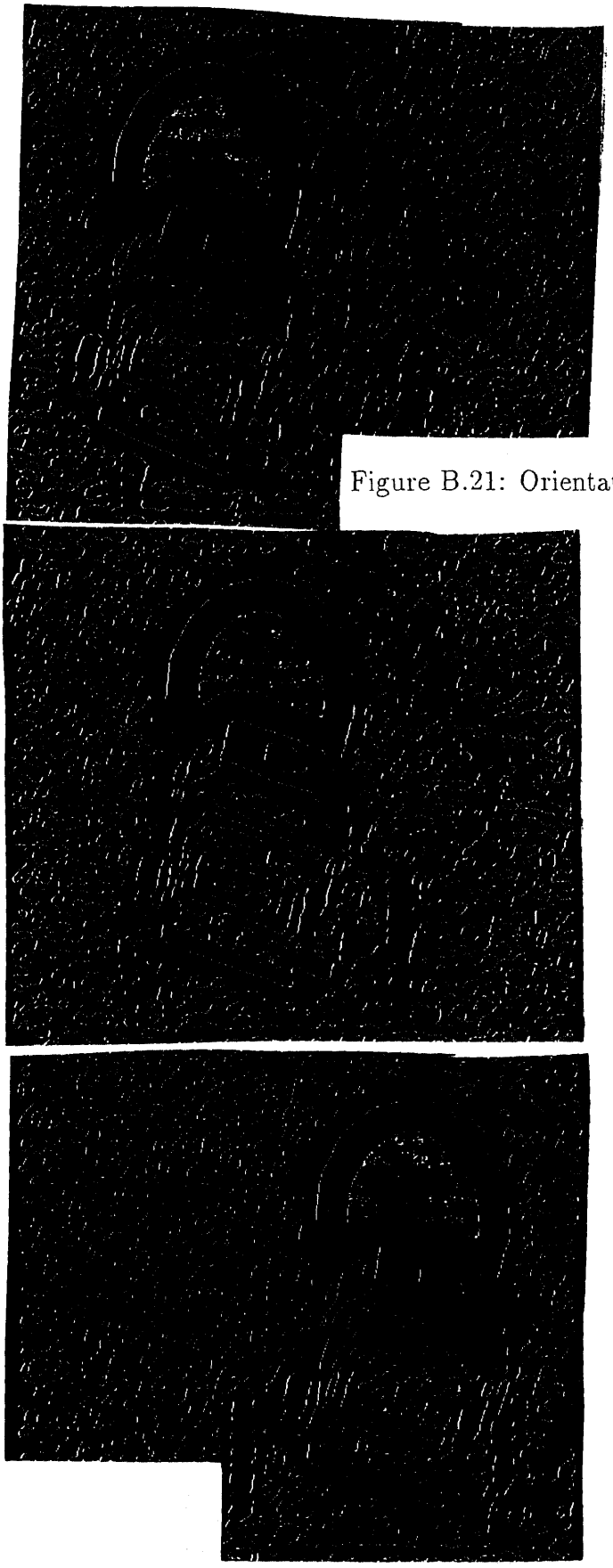


Figure B.21: Orientation of

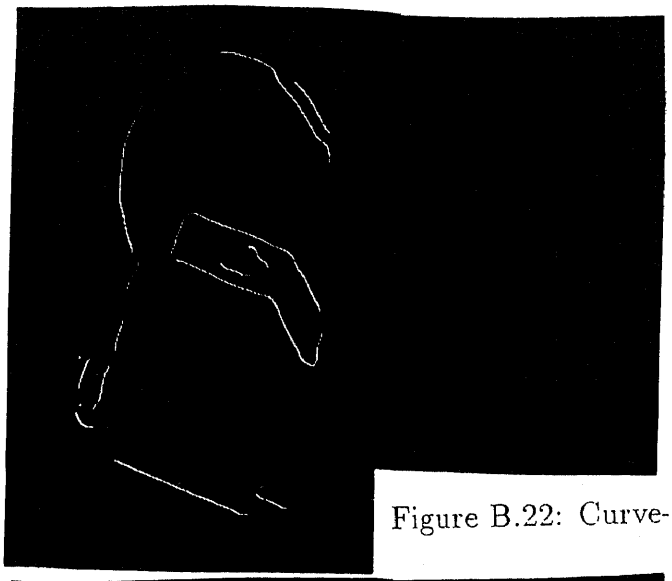
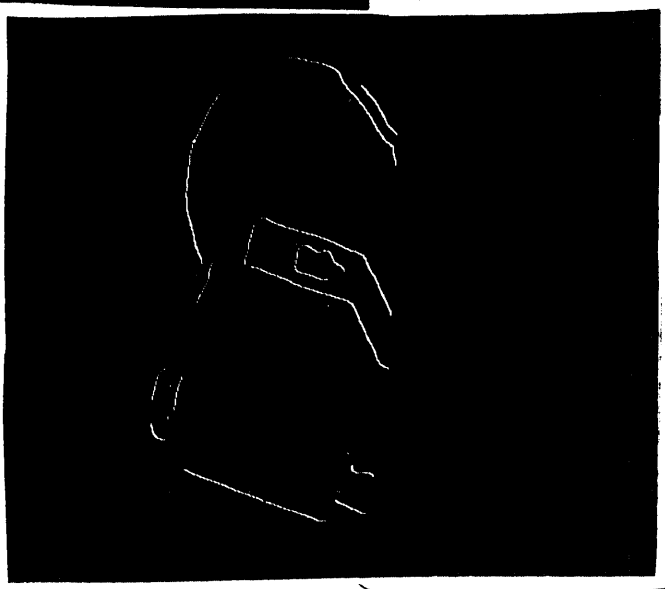


Figure B.22: Curve-Segments of Jack at $\sigma^2 = 20$



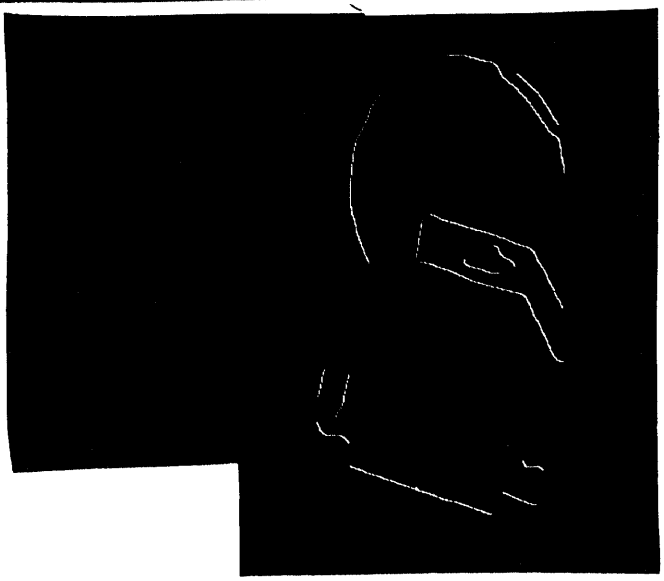
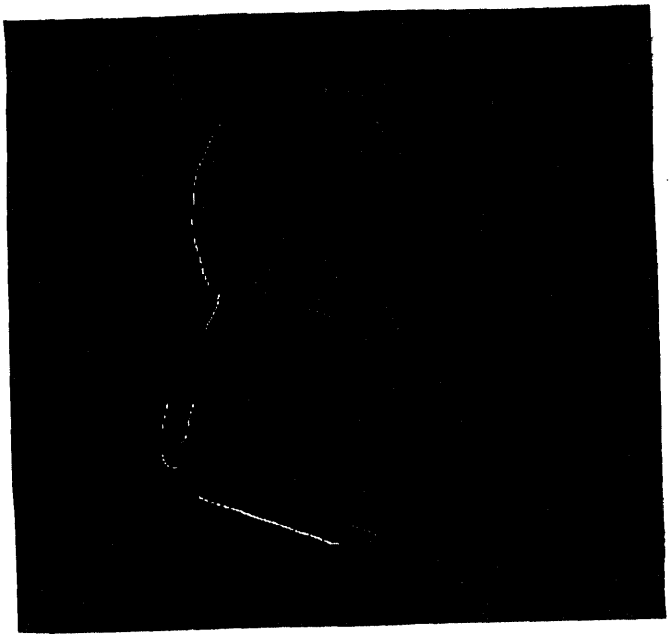


Figure B.23: Disparity Picture of Jack at $\sigma^2 = 20$



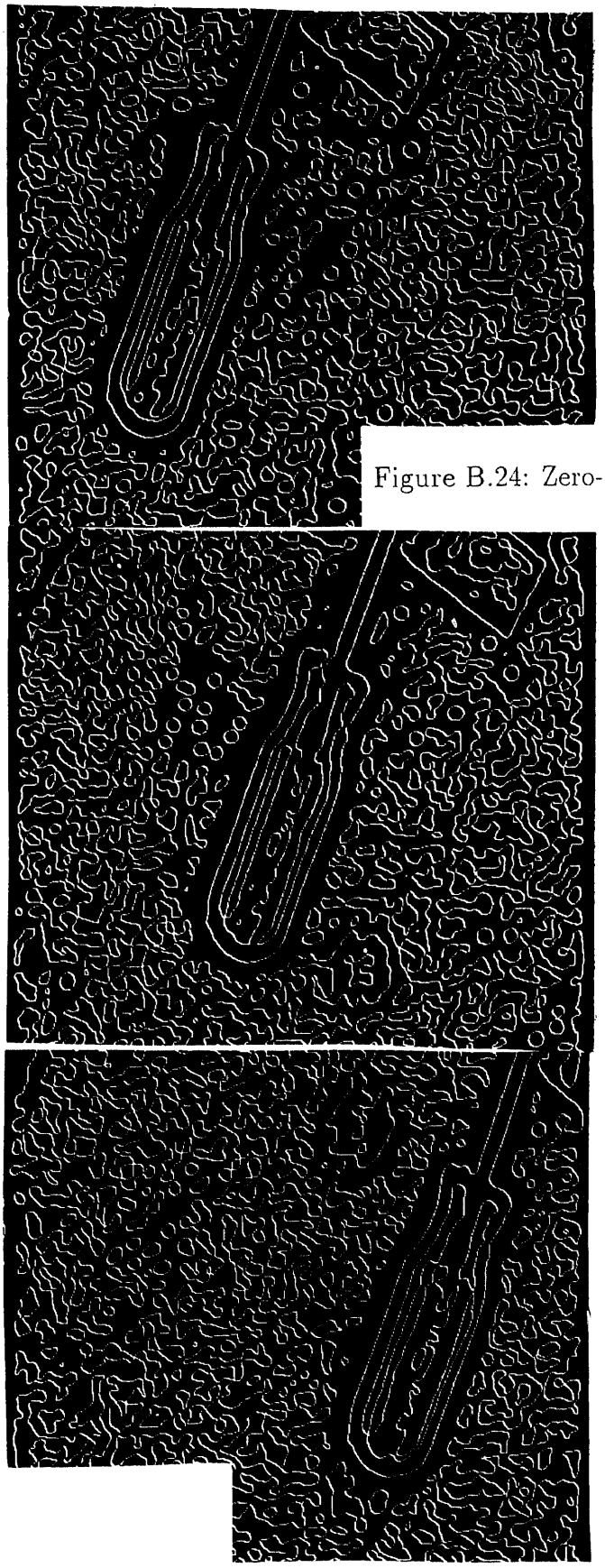


Figure B.24: Zero-Crossings of Screwdriver at $\sigma^2 = 20$

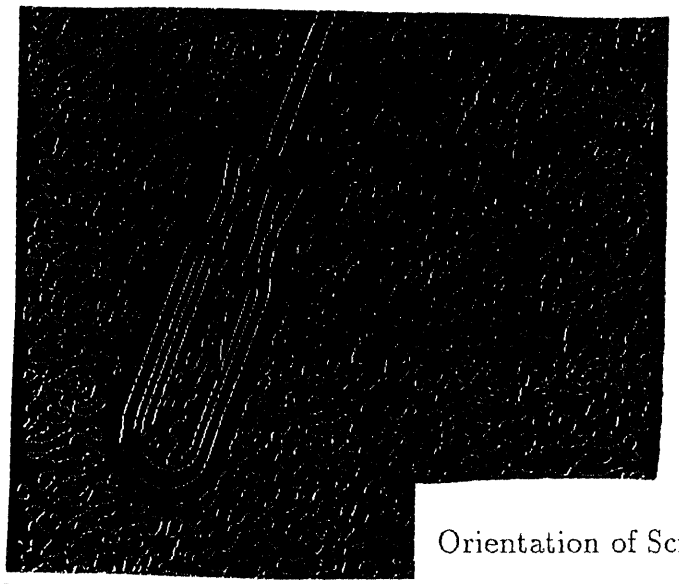
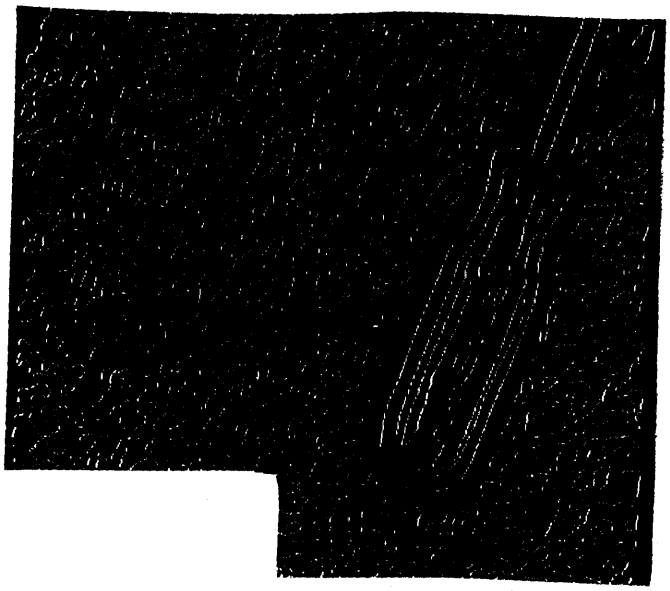


Figure B.25:

Orientation of Screwdriver Zero-Crossings at $\sigma^2 = 20$



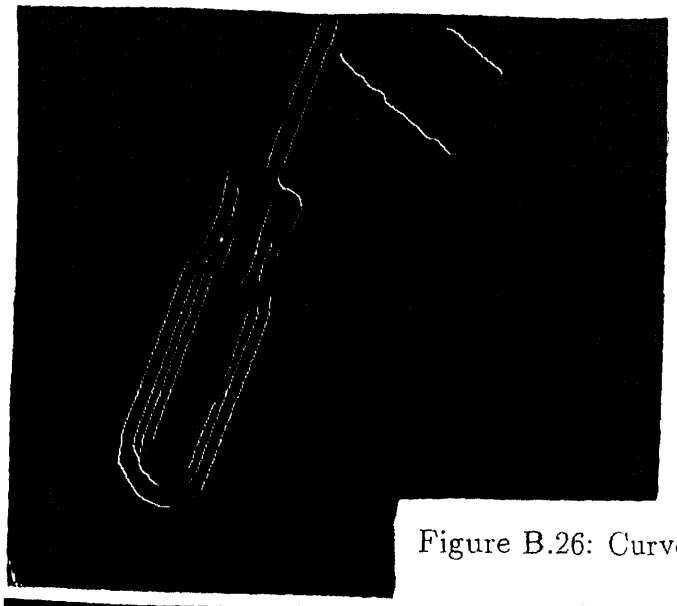


Figure B.26: Curve-Segmnets of Screwdri

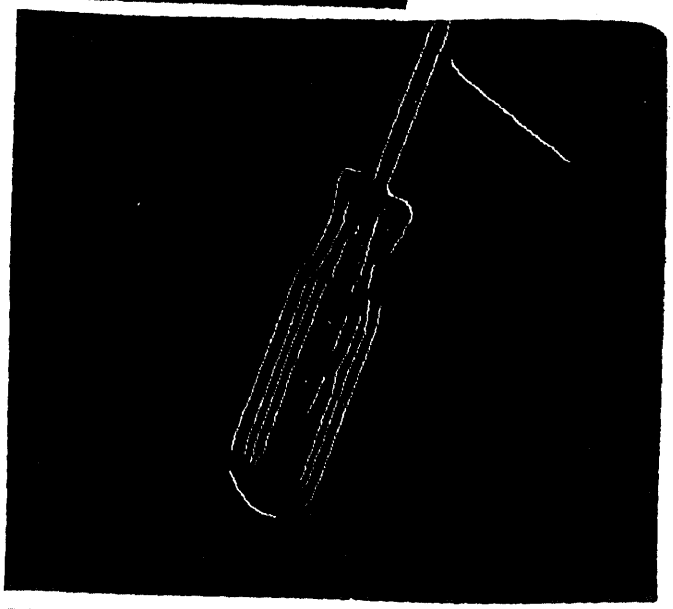
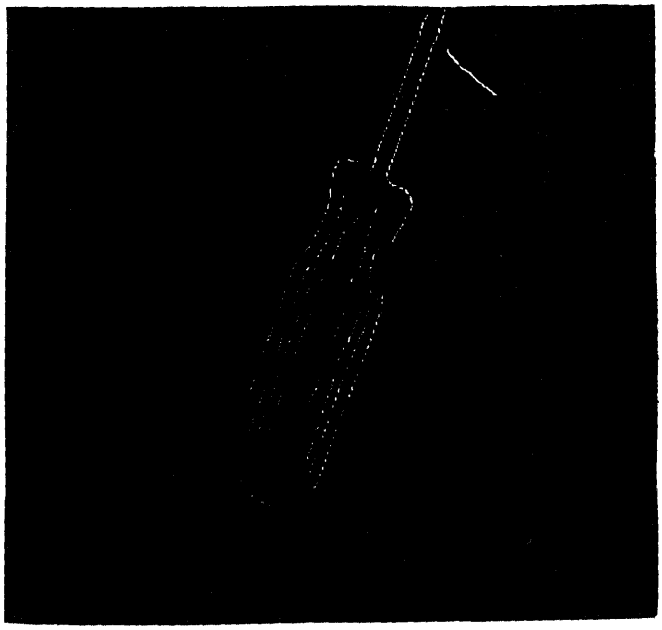


Figure B.27: Disparity Picture of Screwdriver at $\sigma^2 = 20$



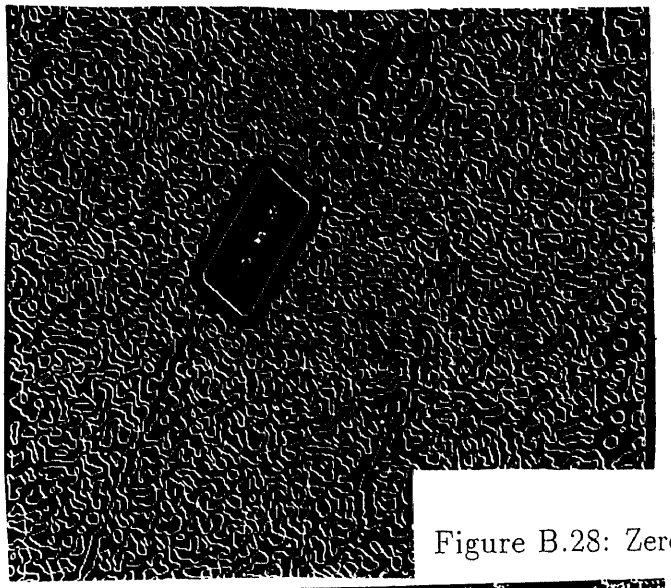
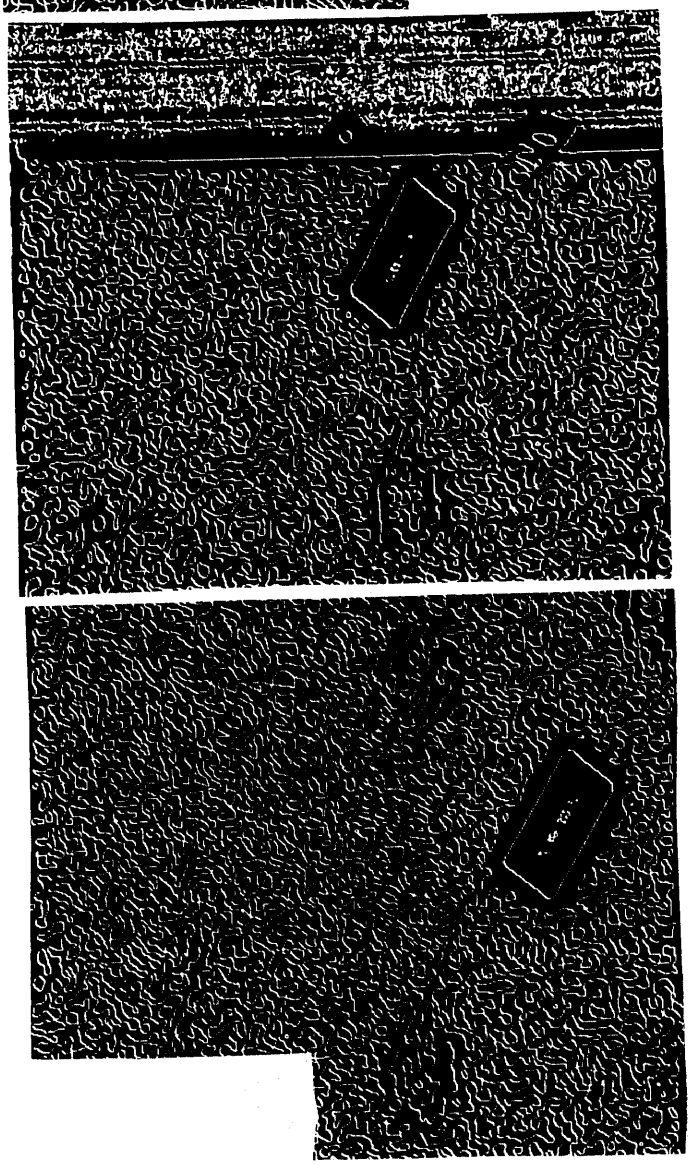


Figure B.28: Zero-Crossings of Tab at $\sigma^2 = 10$



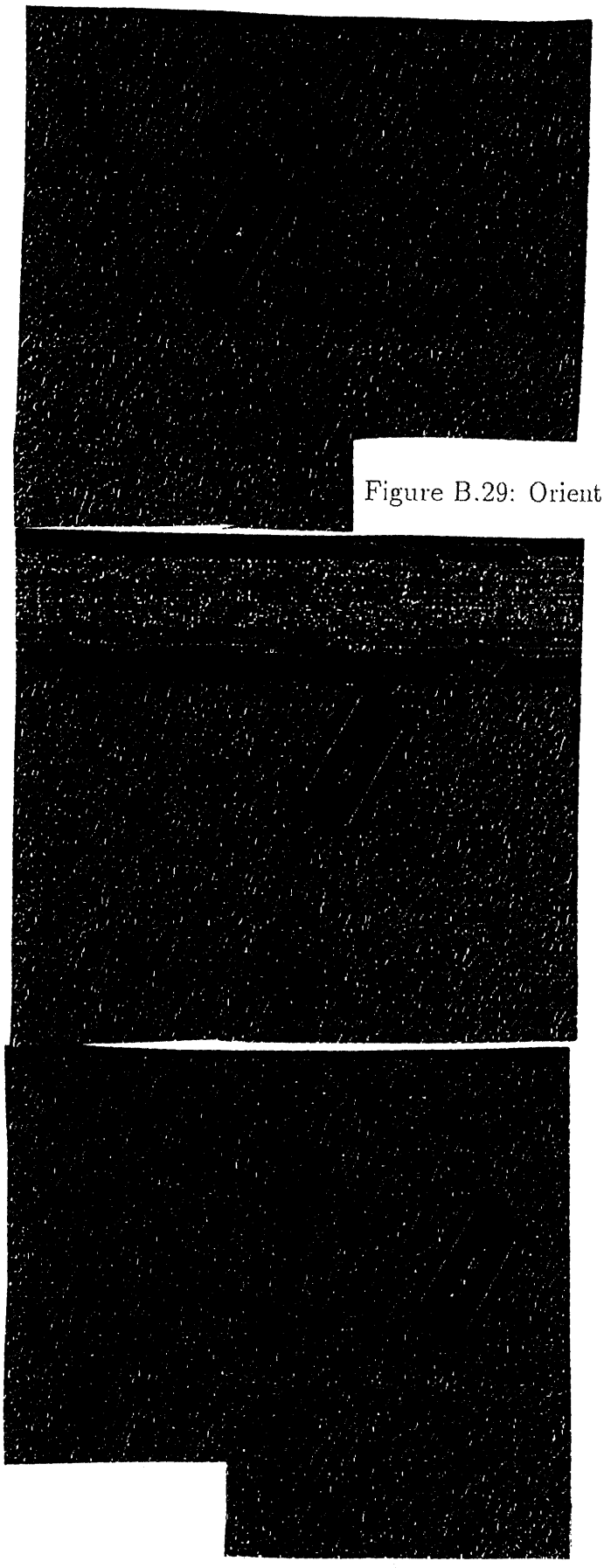


Figure B.29: Orientation of Tab Zero-Crossings at $\sigma^2 = 10$

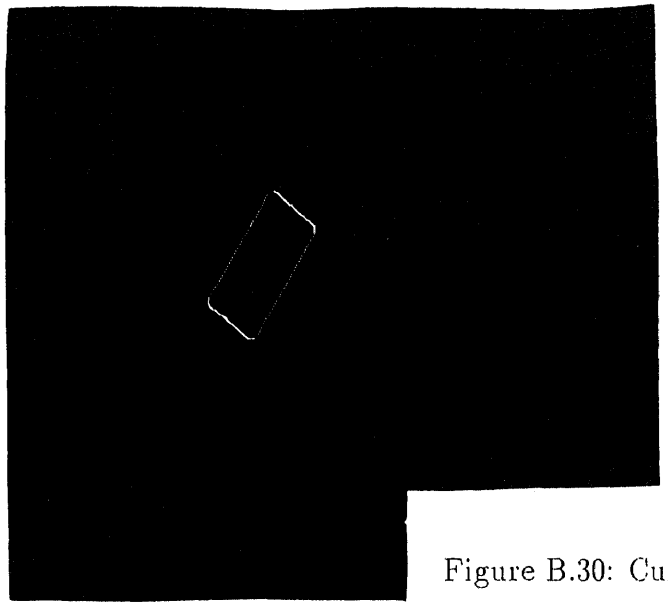


Figure B.30: Curve-Segments of Tab at $\sigma^2 = 10$

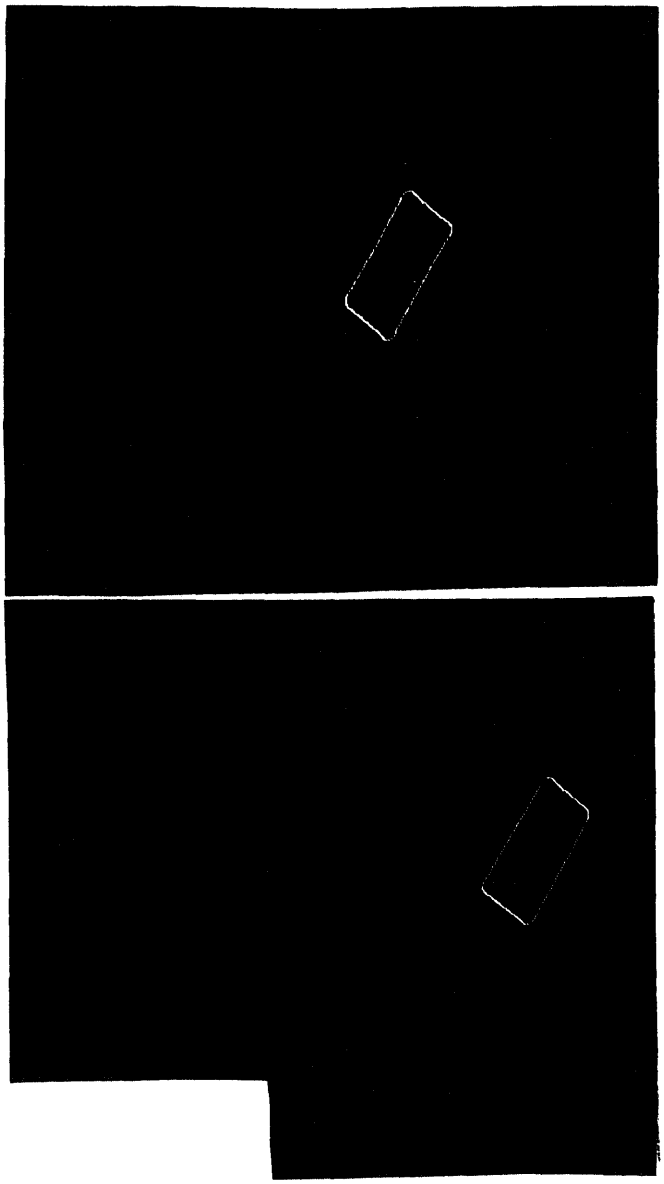
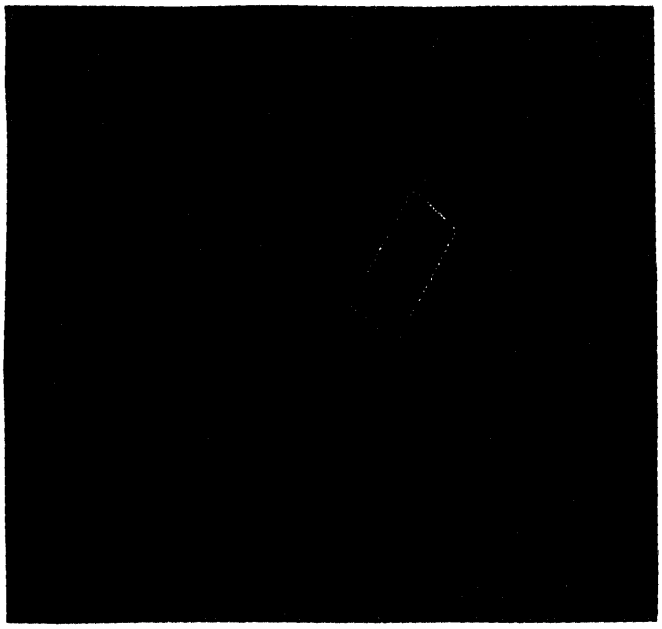


Figure B.31: Disparity Picture of Tab at $\sigma^2 = 10$



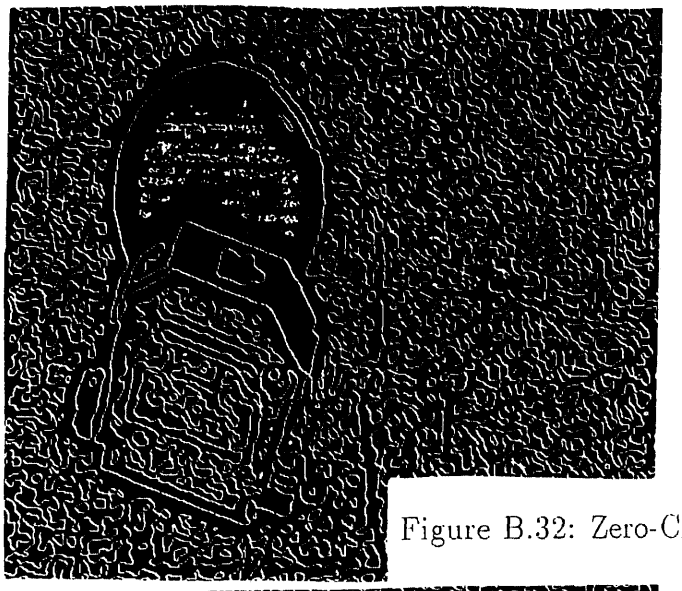
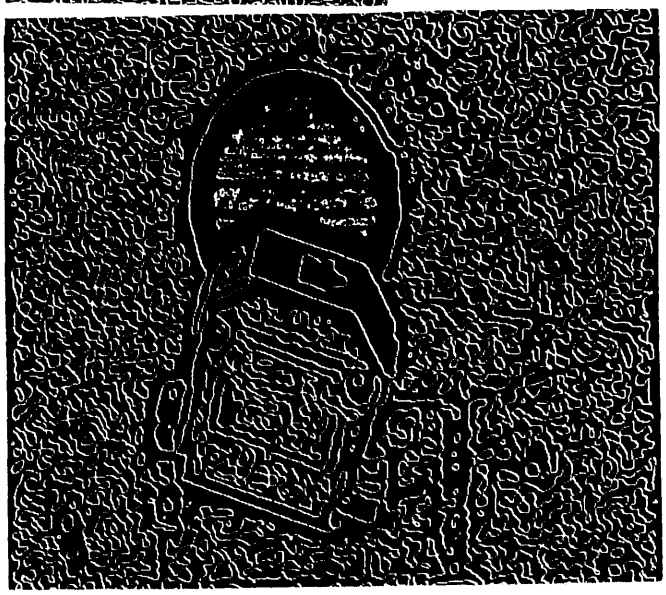


Figure B.32: Zero-Crossings of Jack at $\sigma^2 = 10$



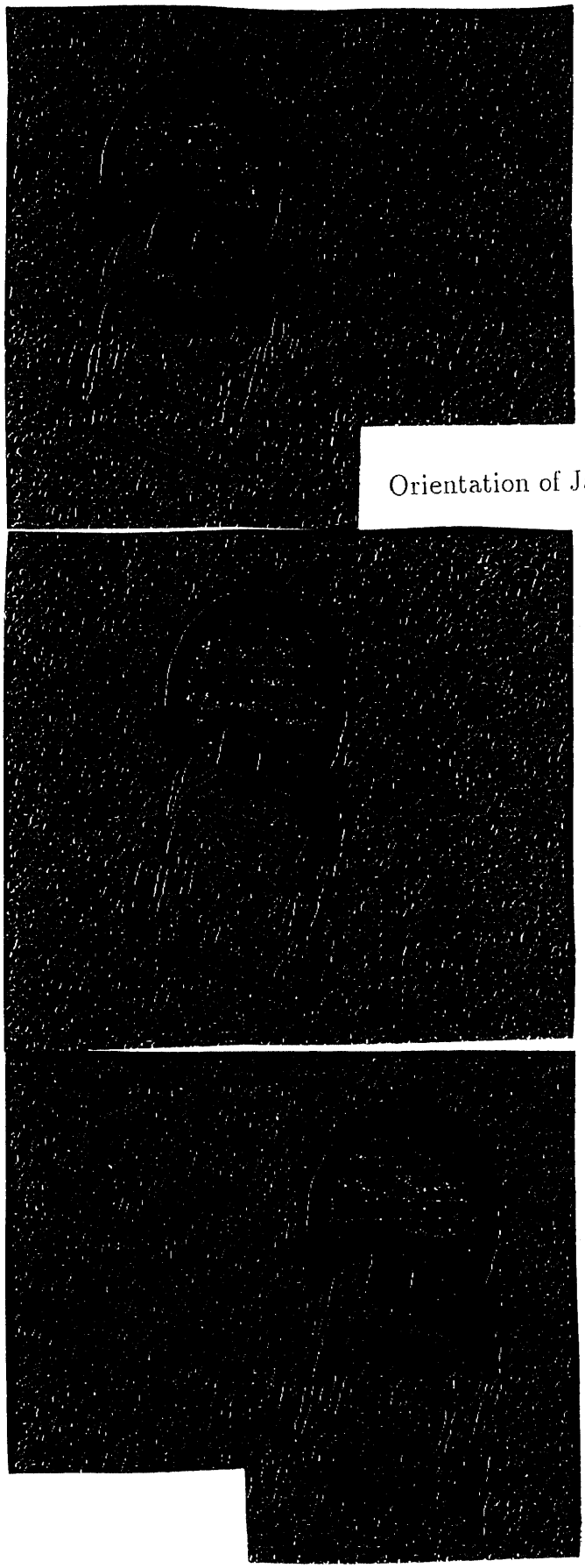


Figure B.33:

Orientation of Jack Zero-Crossings at $\sigma^2 = 10$

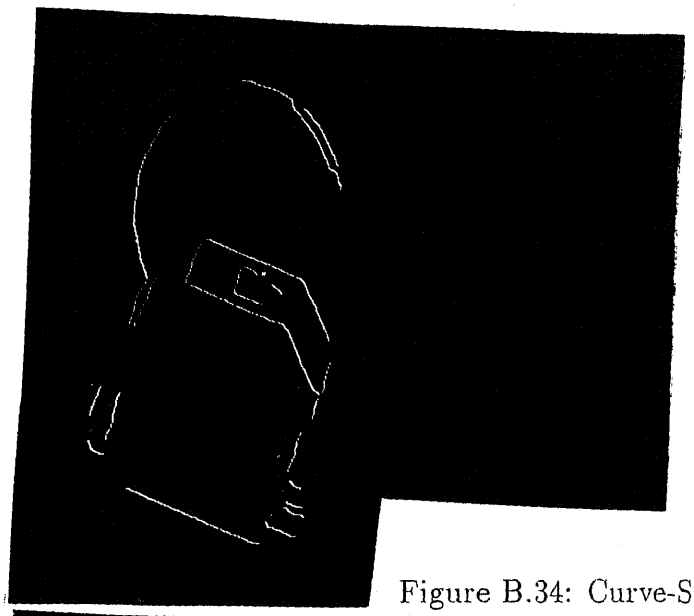


Figure B.34: Curve-Segmnets of Jack at $\sigma^2 = 10$

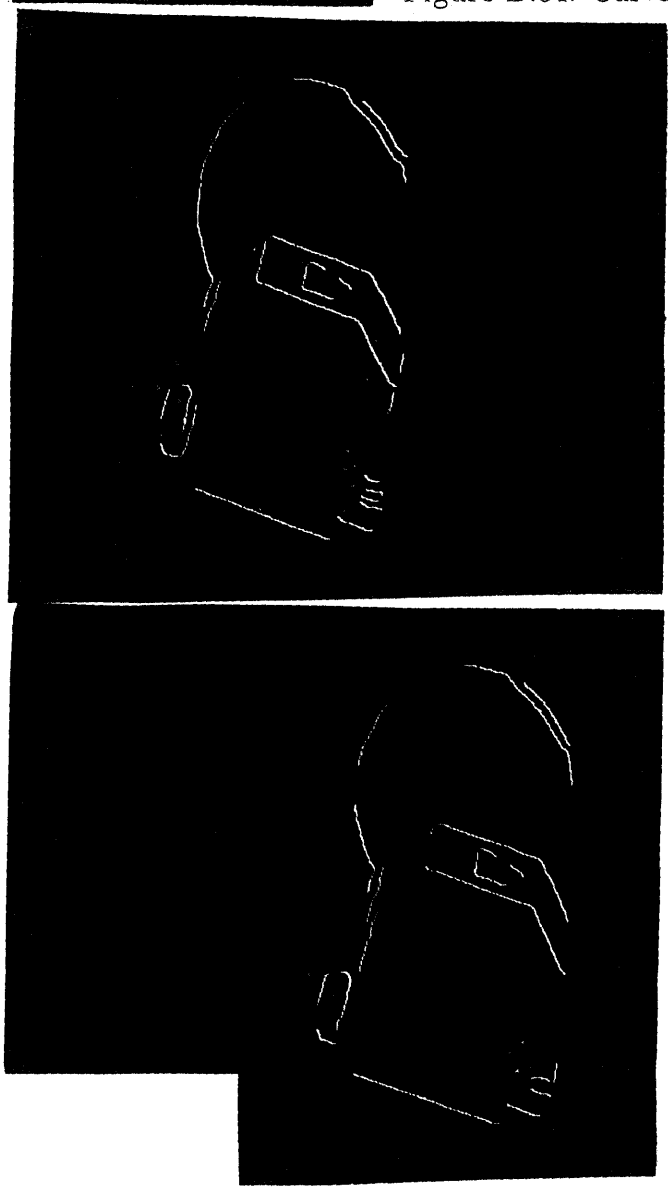
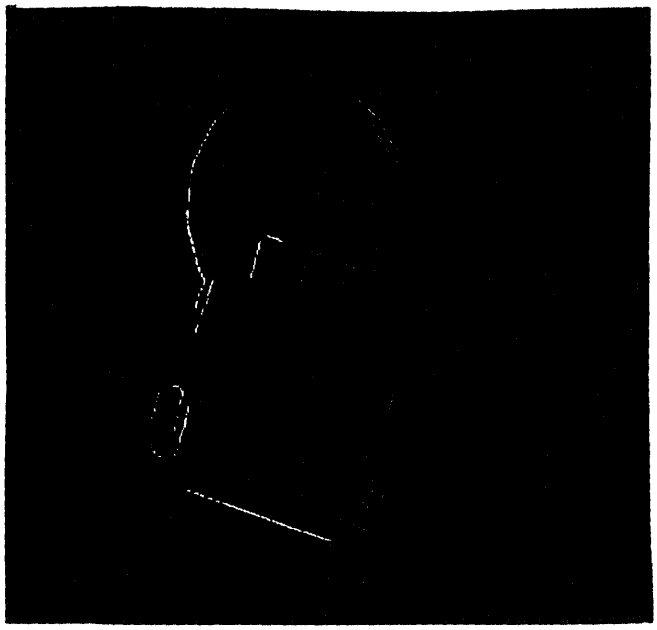


Figure B.35: Disparity Picture of Jack at $\sigma^2 = 10$



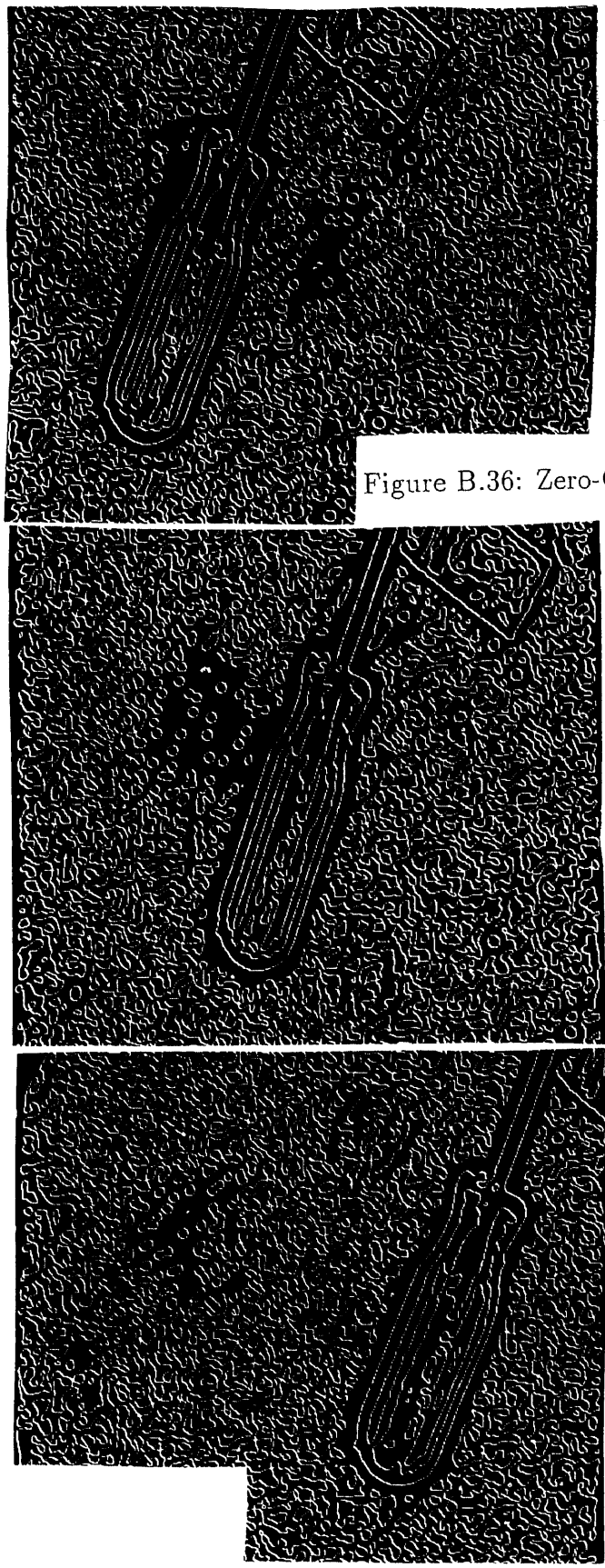


Figure B.36: Zero-Crossings of Screwdriver at $\sigma^2 = 10$

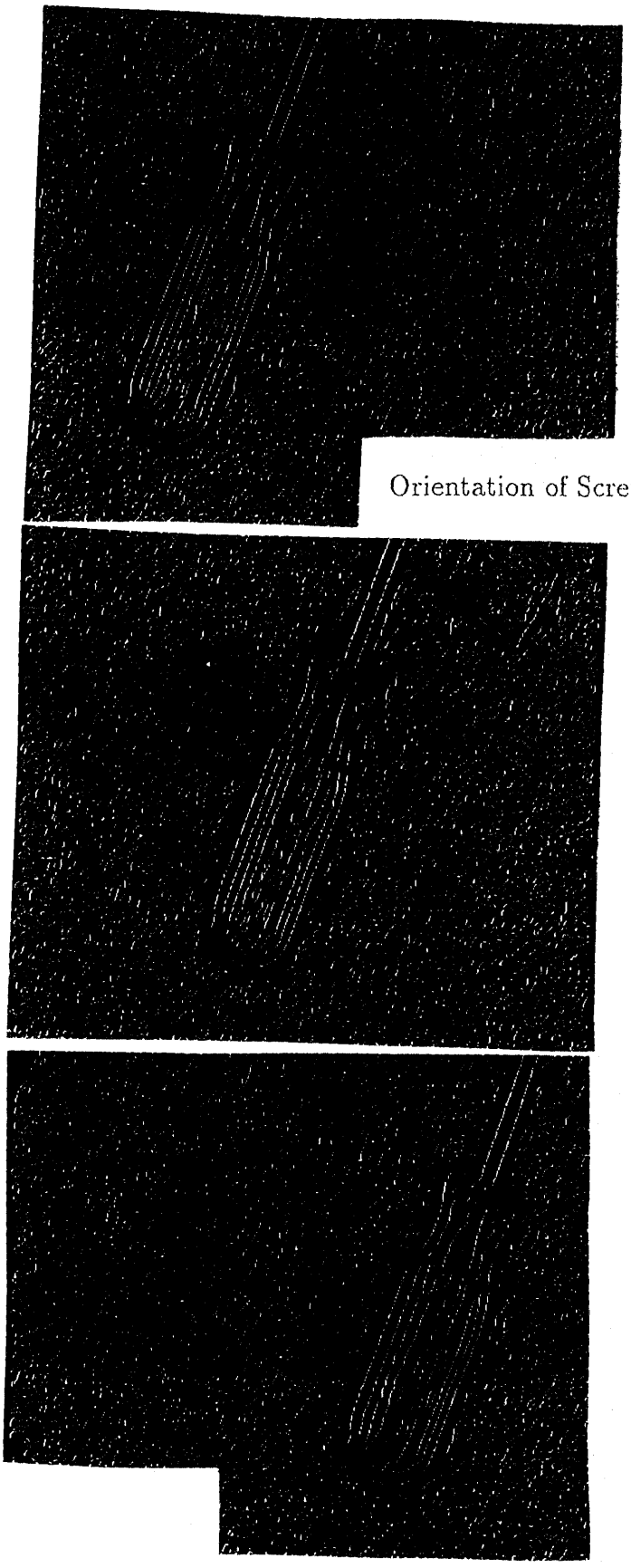


Figure B.37:

Orientation of Screwdriver Zero-Crossings at $\sigma^2 = 10$

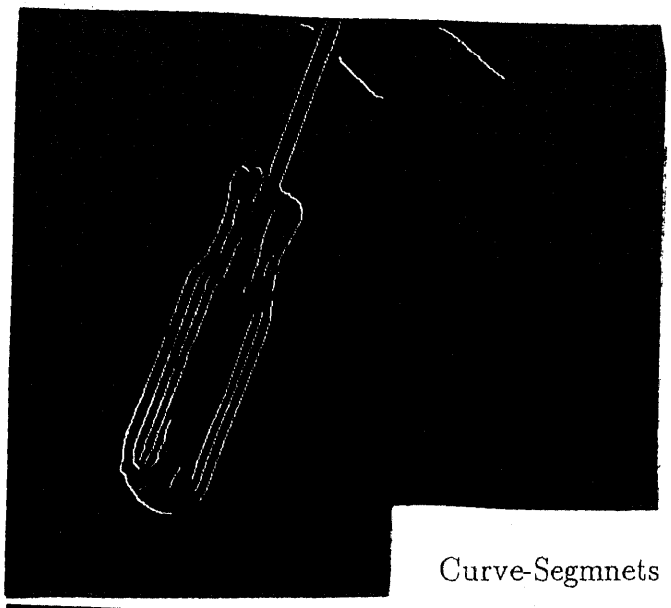


Figure B.38:

Curve-Segmnets of Screwdriver at $\sigma^2 = 10$

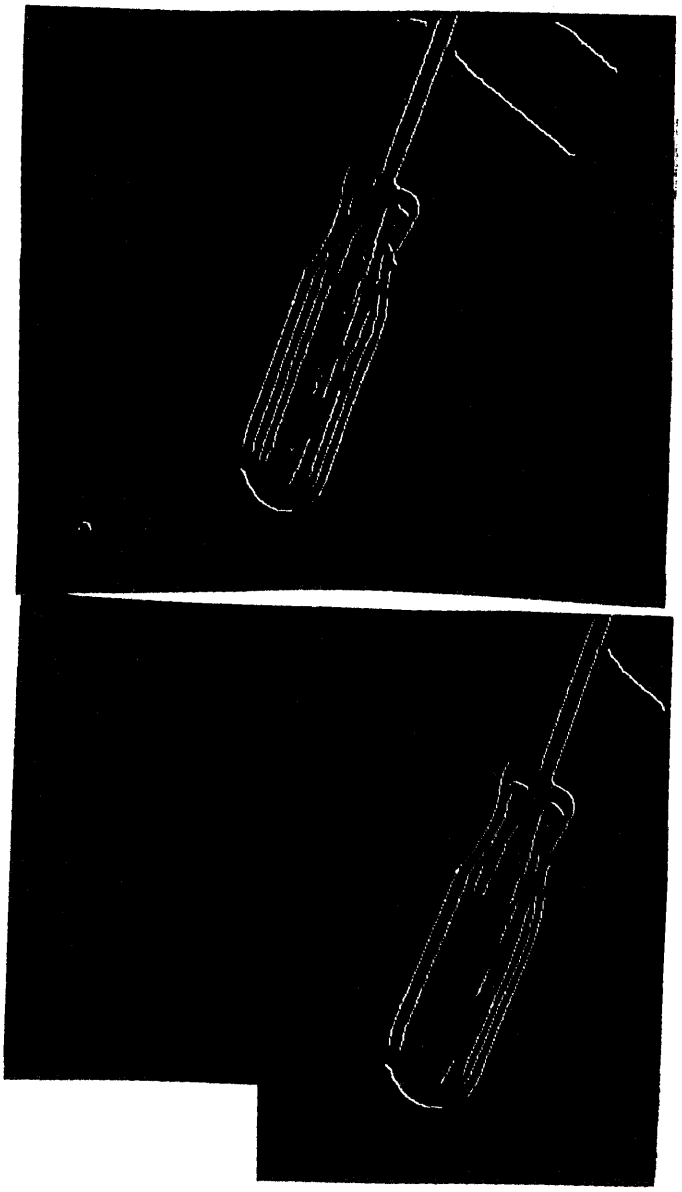
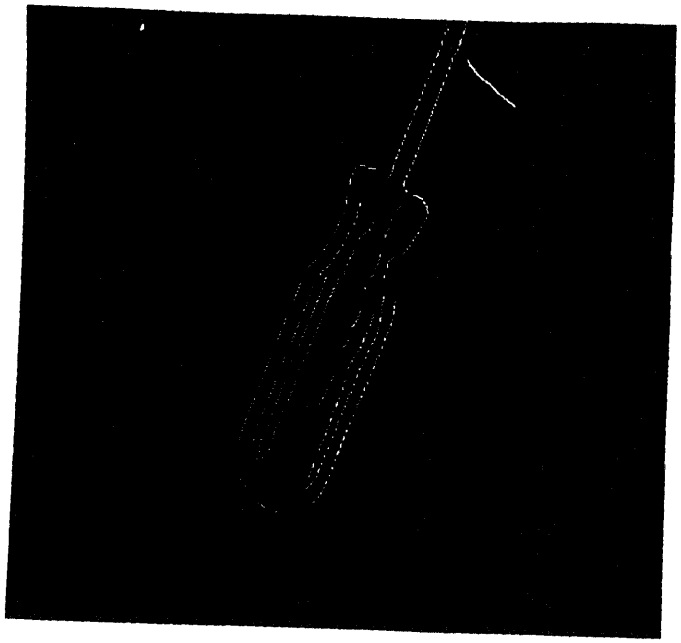


Figure B.39: Disparity Picture of Screwdriver at $\sigma^2 = 10$



Reference

[1] Roger Y. Tsai, " Synopsis of Recent Progress on Camera Calibration for 3D Machine Vision ", Research Report RC 13954(#60041), 1/11/88.

[2] Roger Y. Tsai, "Review fo RAC-based Camera Calibration", Research Report RC 13888(#62428), 8/1/88.

[3] Roger Y. Tsai, " A New Technique for Autonomous and Efficient 3D Robotics Hand/Eye Calibration ", Research Report RC 13212(#59370), 10/20/88.

[4] Wolfgang Nicolin, " An Implementation of Camera Calibration for Computer Vision ", *MSEE Thesis, Department of Electrical Engineering, New Jersey Institute of Technology*, 1989.

[5] Grimson, W.E.L., " From Images to Surfaces: A Computational Study of the Human Early Visual System ", MIT Press 1981.

[6] Grimson, W.E.L., " Computational Experiments with a Feature Based Stereo Algorithm ", IEEE Trans. Pattern and Machine Intell., Vol.PAMI-7, No.1, Jan. 1985.

[7] Marr, D. and Ildreth, E., " Theory of Edge Detection ", Pro. R. Soc. Lond. B207, pp.187-217, 1980.

[8] Marr,D. and Poggio, T., " A Theory of Human Stereo Vision ", MIT Artificial Intell. Memo No.451, Nov. 1977.

[9] Yakimovesky, Y. and Cunningham, R., " A System for Extraction Three Dimensional Measurements from a Stereo Pair of TV Cameras ", Computer Graphics and Image Processing 7, pp.195-210, 1978.

[10] Nasrabadi, N.M., Liu, Y., and Chiang, J., " Stereo Vision Correspondence Using a Multi-Channel Graph Matching Techniques ", IEEE Int. Conf. on Robotics and Automation, April 25-29, 1988.

[11] Haralick, R.M., " Digital Step Edges from Zero-Crossings of Second Directional Derivatives ", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-6, pp.58-68, Jan. 1984.

[12] Torre V., and Poggio, T.A., " On Edge Detection ", IEEE Trans on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No.2, pp. 147-163, March 1986.

[13] Ballard, H.D. and Brown, C.M., " Generalizing the Hough Transform to Detect Arbitrary Shapes ", Pattern Recognition Vol.13, No.2, pp. 111-122, 1981.

[14] Nasrabadi, N.M., Liu, Y., " Application of Multi-Channel Hough Transform to Stereo Vision ", The Advances in Intelligent Robotics System and Computer Vision, SPIE 484, November 1-6, 1987.

[15] Hoff, W., and Ahuja, N., " Extracting Surfaces from Stereo Images: An Integrated Approach ", IEEE, The First International Conference on Computer Vision, pp. 284-294, June 8-11, 1987.

[16] Rosenfeld, A., " Digital Picture Processing ", Academic Press, 1982.