An Algorithm for Detecting Line Segments in Digital Pictures

Abdol-Reza Mansouri

B Eng (McGill University) 1983

Department of Electrical Engineering

McGill University

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Engineering

March 1987

© Abdol-Reza Mansouri

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-38294-5

RESEARCH THESES/REPORTS

THESIS NO. 87-7TM

TITLE: An Algorithm for Detecting Line Segments in Digital Pictures

AUTHOR(S) Abdol-Reza Mansouri

Department of Electrical Engineering

McGill University

Montreal, CANADA

DATE: March, 1987

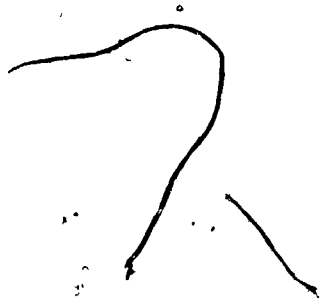
GRANT OR CONTRACT: NSERC

NO. OF PAGES: 72

SUPERVISOR: Professor Alfred S. Malowany

Abstract

This thesis presents an efficient method for detecting straight line segments in digital pictures using a hypothesis prediction/verification paradigm In this paradigm, a straight line segment of predefined length is predicted to exist at some particular pixel location. The orientation of this predicted line segment is based on the edge orientation at the pixel location This prediction is then verified against statistical tests performed on the line As a result, the predicted line is either validated as being a line segment, or it is rejected Non-statistical tests are also developed in order to verify the predicted hypothesis An extension of this algorithm for the detection of lines at different lengths is also presented, and a criterion is defined in order to evaluate the significance of the detected line segments. Finally, the algorithm is successfully tested on a number of different images



Résumé

Cette thèse présente un algorithme pour la detection de lignes dans les images numériques. L'algorithme présenté utilise un processus de prédiction/vérification Plus précisement, un segment de ligne droite est prédit à chaque point de contour dans l'image La longueur de ce segment est specifiée d'avance, tandis que son orientation est déduite de l'orientation du gradient au point de contour consideré Cette prédiction est ensuite verifiée au moyen de tests statistiques et autres sur les échantillons qui constituent le segment en question Une extension de l'algorithme pour la detection de segments de differentes longueurs est presentée et un critère est défini en vue de leur évaluation Enfin, la performance de l'algorithme est démontrée sur nombre d'images réelles

Acknowledgements

Si Paris valait bien une messe, une thèse, elle, vaut bien un discours. Dans ce qui suit, je tiens à exprimer ma reconnaissance à tous ceux qui, de près ou de loin, m'ont encouragé dans la poursuite de mes objectifs.

Tout d'abord, je remercie mes parents, ainsi que ma soeur, pour m'avoir continuellement encouragé dans mes études et pour m'avoir apporté un soutien moral et financier constant

Ensuite, je remercie mon professeur et directeur de thèse, le professeur Alfred S. Malowany pour avoir supervisé ma thèse et pour m'avoir accordé sa confiance, confiance qui s'est traduite par un interêt accru dans le domaine etudié

Je remercie le professeur Martin D Levine pour m avoir initié aux principes de la vision par ordinateur et pour m'avoir encouragé dans la poursuite de cette voie soit par le soutien financier qu'il a bien voulu m'accorder, soit par les discussions qu'il a toujours accepté d'entreprendre pour m'aider dans mes travaux de recherche.

Je remercie le professeur Jorge Angeles pour avoir bien voulu m'accorder l'opportunité de travailler sur une variété de projets interessants et instructifs et pour avoir exprimé un vif intérêt dans mon travail

Je remercie le professeur Steven W Zucker pour m'avoir intéressé à la recherche fondamentale et aux fondements de la vision par ordinateur

Je remercie le Ministère de l'Education de la Province du Québec ainsi que le Conseil National de la Recherche en Sciences Naturelles et en Génie du Canada pour leur assistance financière.

Je remercie tous les camarades que j'ai cotoyé durant ces années d'étude et auxquels je dédie cette thèse. En particulier, je tiens a mentionner Robert Bergevin, Bruno

Bruno Blais et Pierre Sicard, qui ne finiront jamais de m'impressioner; Faycal Kahloun et Stéphane Aubry, mes baroudeurs. Michel Habib et Patrick Aboussouan, mes compagnons de toujours; De plus, je tiens a remercier Dominic Chau, Wade Hong, Tat-Chi Chung, Baris Demir, Guy Godin, Christian Consol, Nemri Chafye, Kamal Gupta, Stavros Mohlulis, Allan Dobbins, Mike Sabourin, Pierre Parent, Peter Sander, Iskender Paylan, Gracy Mimran, Moshe Cohen, Naseer Ali, Roger Hernandez, Norah Link, Chantal David, Gregory Carayannis, Paul Freedman, Christian Michaud, Daniel Kornitzer, Frederic Leymarie, Lee Iverson, et tous ceux que je risque d'avoir oublié de mentionner, pour avoir agrémenté mon séjour à McGill, ainsi que Mike Parker et Cem Eskenazi pour m'avoir depanné à plus d'une reprise Je remercie aussi Frank Ferrie, John Studenny et Yvan Leclerc pour toutes les discussions interessantes et instructives

A tous, un grand merci

Contents

List of Figures

Chapter 1 Introduction

Computer vision is the field of image interpretation by computer Images are formed of picture elements or pixels and are the projection of 3-D scenes onto a 2-D field The main problem adressed is therefore how do we interpret images acquired by external sensors ? The final goal of this interpretation stage, therefore, is to arrive at a symbolic representation of the image in terms of known elements

One could view the process of image interpretation as consisting of one gigantic monolithic stage. The input to this stage would be the raw image and the output would consist of symbolic descriptions of the content of the image. Considering computer vision in this manner does not help all of the complexity of the interpretation task is embedded in one black box where only its input-output behavior is known The alternative way of looking at computer vision is to do so in a modular way rather than viewing the interpretation step as one single step, we decompose it into a number of simpler operations Thus, we have a cascade of processing elements, where the output of one feeds the input of the next. Starting from the raw image data therefore, the final symbolic representation of the image is achieved through processing a number of intermediate representations (see figure 1.1).

Since the processing elements are dependent on the symbolic representations chosen, the question that should be asked at this point is: what are the intermediate representations that need to be chosen? In the applied vision context, the intermediate

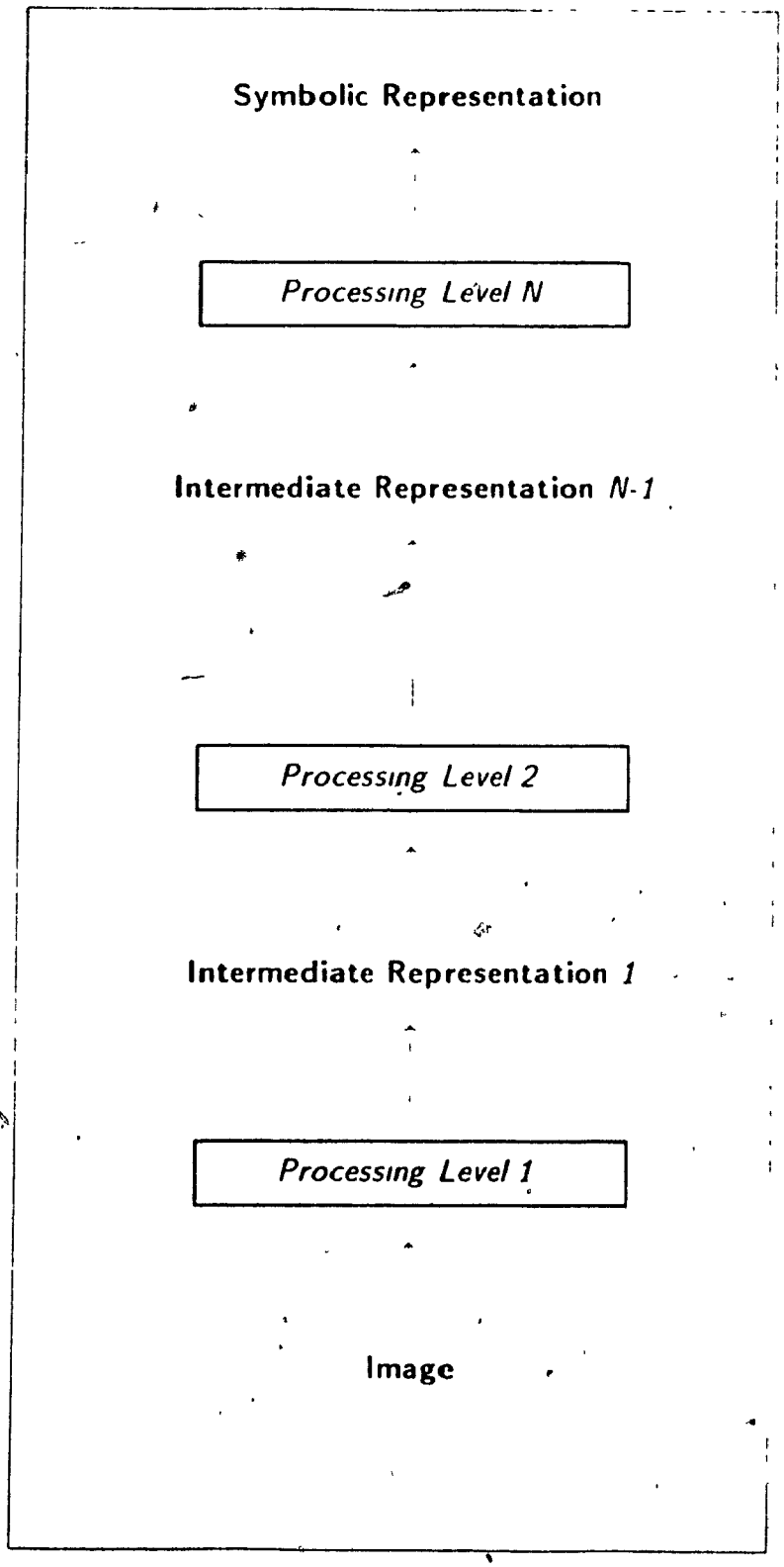


Figure 1.1 Modular processing

representations are problem dependent, and representations which might be suitable for

interpreting one class of images might not be appropriate for interpreting another.

Another aspect of this cascade-type of processing is the flow of control In this context, flow of control refers to the interdependencies between the different processing elements These interdependencies can be roughly classified into two categories: top-down and bottom-up In bottom-up control, each processing element processes the data from the previous (lower) level in the hierarchy, and feeds the next (higher) level In this paradigm, the data flows from the image up to its symbolic representation, and at each level, no knowledge is assumed about higher levels Thus, each processing element is as general purpose as possible and is not biased by what higher level processing dictates This is in direct contrast to top-down control where in addition to feeding the next (higher) stage in the hierarchy, a processing element can feed information back to the previous (lower) level in the form of problem-domain knowledge and heuristics In top-down control, therefore, each processing element is guided by the expectations of higher level modules and hence loses its generality by being problem dependent

Choosing the nature of the different intermediate representations and of the control scheme is not a simple task when addressing the vision problem in general In applied vision however, the scene to be interpreted can be constrained to belong to a number of classes This thesis presents an algorithm for computing one of these intermediate level representations in an applied vision context and without any a priori knowledge of the scene content. More specifically, this thesis presents an algorithm for detecting line segments in gray level digital images without any a priori notion of the higher level structures that they might constitute (e g rectangles, polygons,). Why lines and not circles ? Simply because in most industrial vision tasks, linear features are predominant, and the main requirements of any processing stage are that it be robust to variations in the input data and be computationally efficient The algorithm presented here could therefore be used as one building block of a complete image interpretation system An example such system is shown in figure 1.2 for the case of detecting polygonal objects.

The problem to be solved is then the following: given a gray level image, find

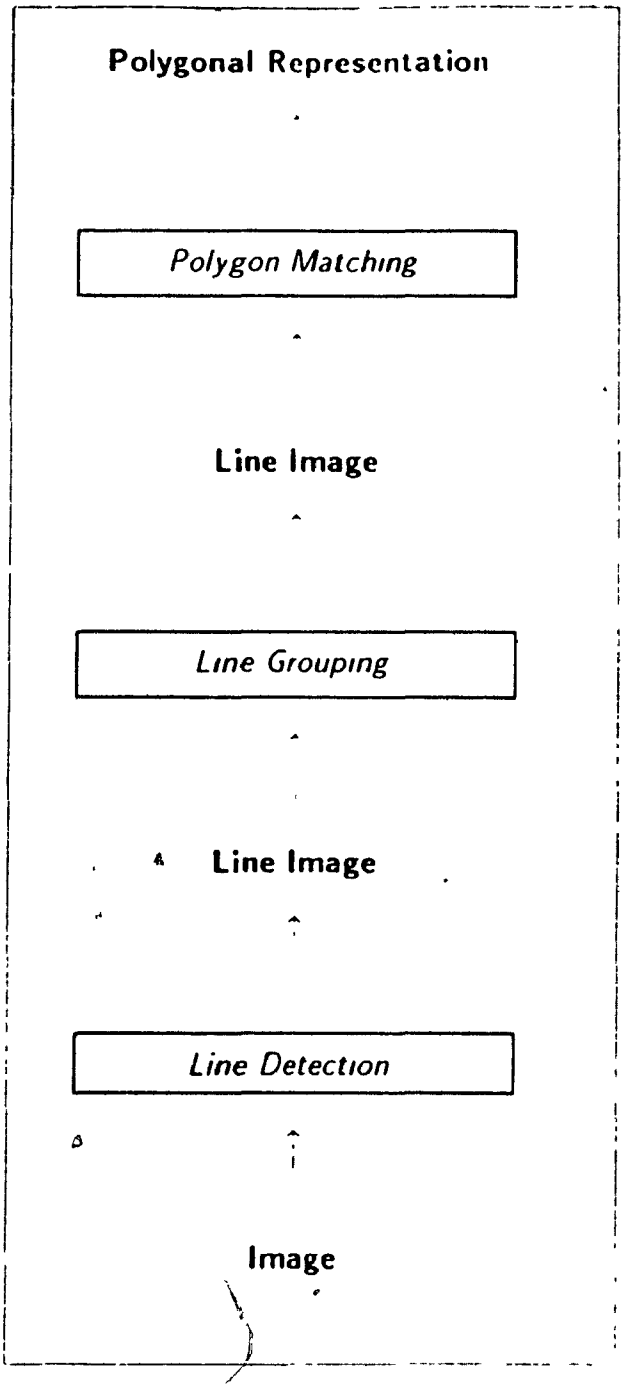


Figure 1.2 Example bottom-up system for detecting polygonal objects

the lines in this image As will be seen later, we will restrict ourselves (without loss of generality) to the detection of contour lines, since we believe this is the most interesting case in practical applications The solution we present is based on the efficient hypothesis prediction/verification paradigm It consists of predicting the presence of a line at a possible pixel location, and then verifying this assertion through statistical tests The efficiency of

this framework and the simplicity of the tests involved contribute to a large extent to the performance of this algorithm, and experiments that have been performed show that is both robust to fluctuations in the input data and computationally efficient

This thesis is structured as follows: chapter 2 provides an overview of algorithms for four different approaches to line detection, chapter 3 presents the proposed algorithm, chapter 4 discusses experimental results, and finally, in chapter 5, appropriate conclusions are drawn and possible extensions to the proposed algorithm are suggested

Chapter 2 Line Detection: a Survey

2.1 The Hough Transform

One of the earliest algorithms which was developed for line detection is the Hough transform[1][2] The Hough transform is a mapping from the image space into a parameter space in which shape features (in this case lines) are more explicit Consider an image function f defined over a discrete two-dimensional domain defined by the integer variables x and y It is assumed that f defines a binary image (i e $f(x,y) = 0$ or $f(x,y) = 1$) as a result of preprocessing, and that the lines to be detected are formed of picture elements (x,y) such that $f(x,y) = 1$ In other words, the background has intensity 0 and the lines have intensity 1

One of the earliest parameterizations that was performed for the Hough transform was the slope-intercept parameterization In this case, a picture element (x,y) is deemed located on a line with slope m and intercept n if and only if $f(x, mx+n) = 1$ The Hough transform then consisted of a mapping of the image space into the parameter space formed by the range of values of the parameters m and n Thus, for each point (x,y) in the image space that has the property that $f(x,y) = 1$, the line $n = y - mx$ is formed in the parameter space $m - n$. Thus, the Hough transform maps each point in the image space into a line in this parameter space. The interesting property of the Hough transform is that collinear points in the image space are mapped into parameter space lines which intersect

at exactly one point. Conversely, the number of intersecting lines at an intersection point (m, n) in parameter space corresponds to the number of collinear points in image space which are located on a line with slope m and intercept n The implementation of this algorithm, which was proposed by Duda and Hart[2], goes as follows, set up an accumulator array (parameter space) and for each point (x, y) in the image such that $f(x, y) = 1$, increment the accumulator cells $(m, y - mx)$ for all values of m Thus, after the transform is applied, intersection points are recognizable by their correspondingly large count in the accumulator array In addition to this implementation scheme, Duda and Hart proposed a reparameterization of the lines in terms of the distance and orientation of a line with respect to the image coordinate system The reason for doing so is that the values of m and n could grow unbounded for certain line configurations With the new parameterization, a point (x, y) is located on a line defined by a distance ρ from the origin and an orientation θ with respect to the $x-$axis if and only if $f(x, y) = 1$ and $\rho = x cos\theta + y sin\theta$ Thus in this case, the Hough transform maps each image feature point (x, y) into a sinusoidal curve defined by the preceding equation. Again, collinear image points yield intersecting curves in parameter space (see figure 2 1)

The Hough transform is a special and discretized version of the Radon transform[3], the latter being a mapping from the space of functions defined on the image space into the space of functions defined on the parameter space Thus assuming an arbitrary image function f defined on some domain D of $\mathbf{R}^2$ (the continuous image), the Radon transform of f associated with a line L of the plane is given by

$$\check{f} = \mathcal{R}f = \int_L f(x, y)ds \tag{2.1}$$

with ds being an increment of length along the line L Now consider a line L specified by a distance ρ from the origin and an orientation θ with respect to the $x-$coordinate axis The value of the Radon transform for parameters ρ and θ is then

$$\check{f}(\rho, \theta) = \int_{L(\rho,\vartheta)} f(x, y)ds \tag{2.2}$$

Now let the unit vector $\vec{\varepsilon}$ be defined by

$$\vec{\varepsilon} = \begin{pmatrix} cos\theta \\ sin\theta \end{pmatrix} \tag{2.3}$$

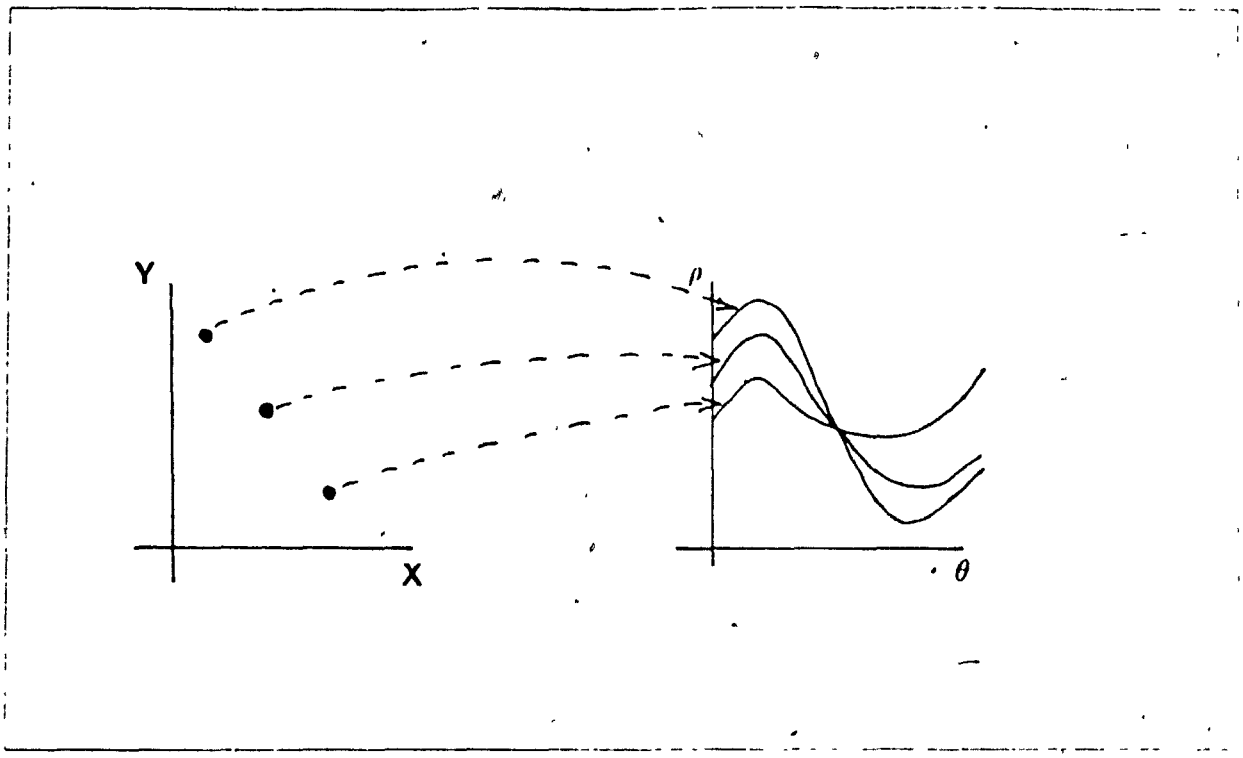


Figure 2.1 Hough transform

and the unit vector $\vec{\epsilon}^{\perp}$ orthogonal to it by

$$\vec{\epsilon}^{\perp} = \begin{pmatrix} sin\theta \\ cos\theta \end{pmatrix} \tag{2 4}$$

$\vec{\epsilon}$ and $\vec{\epsilon}^{\perp}$ are unit vectors respectively orthogonal and parallel to the line L (see figure 2.2).

With this new parameterization, the Radon transform becomes

$$\check{f}(\rho, \vec{\epsilon}) = \int_{-\infty}^{+\infty} f(\rho\vec{\epsilon} + t\vec{\epsilon}^{\perp})dt \tag{2 5}$$

Now consider the function g such that

$$\check{g}(\rho, \vec{\epsilon}) = \int_{-\infty}^{+\infty} f(\rho\vec{\epsilon} + (t + a)\vec{\epsilon}^{\perp})dt \quad a \in \mathbf{R} \tag{2.6}$$

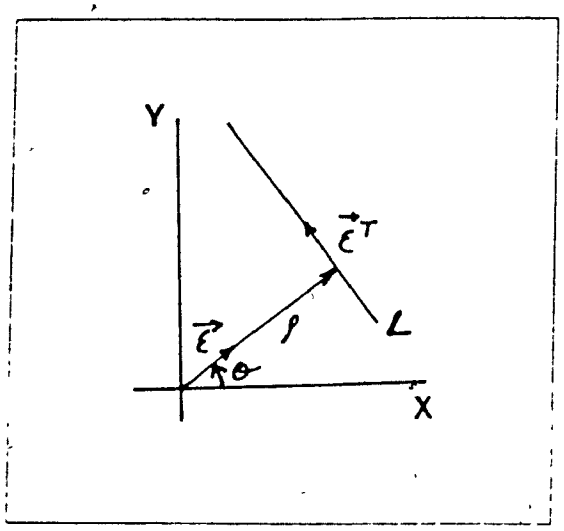


Figure 2.2 Radon transform parameters

In other words, the function g defines a line similar to f in distance and orientation, but different in position (due to the shifting term a). The Radon transform of g is then found to be

$$\begin{aligned}\check{g}(\rho,\vec{\varepsilon}) &= \int_{-\infty}^{+\infty} f(\rho\vec{\varepsilon} + (t+a)\vec{\varepsilon}^{\perp})dt \\ &= \int_{-\infty}^{+\infty} f(\rho\vec{\varepsilon} + (t+a)\vec{\varepsilon}^{\perp})d(t+a) \\ &= \int_{-\infty}^{+\infty} f(\rho\vec{\varepsilon} + t\vec{\varepsilon}^{\perp})dt \\ &= \check{f}(\rho,\vec{\varepsilon})\end{aligned} \tag{2.7}$$

Thus f and g have similar Radon transforms. Similarly due to the linearity of the Radon transform,

$$h(x,y) = f(x,y) + g(x,y) \Rightarrow \check{h} = \check{f} + \check{g} \tag{2.8}$$

What do these results imply? The first result implies that the Hough transform of a line is the same no matter where it is located and as long as its distance to the origin and its orientation are fixed. Thus, position information is not maintained in the

Hough transform. The second result implies that line segments which are not connected contribute to the same cell in parameter space, as long as they have the same parameters ρ and θ. Using these two results, we conclude that in the limit, the Hough transform cannot distinguish between a line of a certain length and a group of collinear but disconnected points.

In addition to these problems which are inherent to the Hough transform, another major problem still exists and is due to the implementation This problem is that of parameter-space quantization A very fine quantization of the parameters ρ and θ will tolerate only very small deviations from collinearity, while a gross quantization will not discriminate between non-collinear points The first artefact will result in clusters of small values in the accumulator array, while the second will result in large peaks

We thus conclude that the Hough transform does not fully solve the problem of line detection. It only consists of a mapping which highlights collinearities The next logical step after the Hough transform is one of *interpretation*: the lines in the image space must be *inferred* from the values of the parameter space In addition to the above problems, the computational complexity of the Hough transform does not work to its advantage Given Φ possible quantized values of the orientation and N feature points, the preceeding equation is to be computed $N\Phi$ times In order to reduce this computational complexity, a scheme that is often used is to convolve the image with a set of orientation-sensitive masks and map each point in image space into only a portion of the curve in parameter space, centered at the pixel's preferred orientation. By using local information derived from the convolution masks, this scheme reduces to some extent cross-interferences between non-collinear line segments.

The Hough transform is a global scheme for line detection and highlights collinearities. Lines, however, are formed of points which possess a precise local structure (e.g. connectedness, location of endpoints) Thus, for a line detection scheme to be successful, this local information must also be taken into account. In what follows, an algorithm which uses this local information in order to enhance lines is presented.

2.2 Relaxation Labelling

Relaxation labelling belongs to a class of iterative algorithms where globally consistent solutions are achieved using local computations[4]. More specifically, for the case of line detection, consider an image as being a graph, with each node in the graph corresponding to a pixel in the image A point in the image can either be part of a line or not In the former case, a certain orientation is associated with the point, which corresponds to the orientation of the tangent to the curve at that point In the latter case, however, no such tangent exists Returning to the graph analogy, each node (i e each pixel) is assigned a set of labels λ (orientations) which are nothing but symbols with a precise semantic meaning Associated with each label λ of a node i is a certainty $p_i(\lambda)$, which reflects the confidence that the label which corresponds to node i is label λ (see figure 2 3)

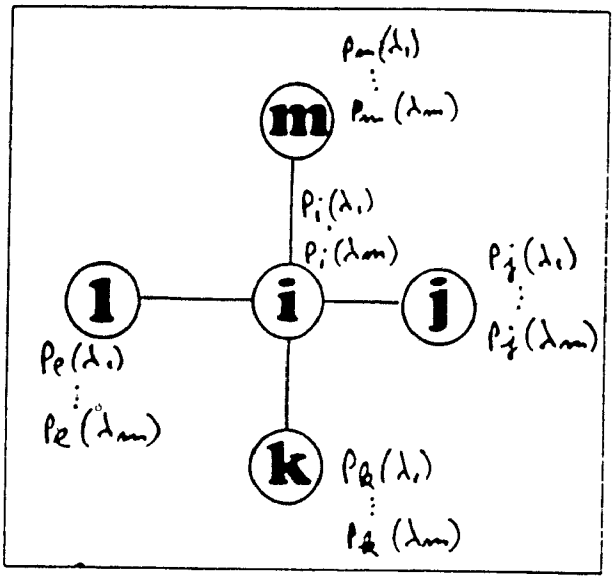


Figure 2.3 Graph labelling

These certainties are normalized by constraining them to add up to 1 and hence

$$\sum_{\lambda=1}^{m} p_i(\lambda) = 1 \qquad (2.9)$$

In addition to this initial label assignement, the interactions between neighboring nodes in the graph have to be defined These interactions are defined through what are called "compatibility functions" The compatibility function of label λ at node i and label λ' at node j specifies how much λ and λ' favor each other when they are at nodes i and j respectively This compatibility function is usually denoted by $r_{ij}(\lambda, \lambda')$ The support given to label λ at node i by label λ' at node j is equal to $r_{ij}(\lambda, \lambda')p_j(\lambda')$ In other words, the compatibility function is itself weighted by the confidence we have in the presence of label λ' at node j The support given by node j to label λ at node i is equal to the total of each individual label support and is hence equal to

$$\sum_{\lambda'=1}^{m} r_{ij}(\lambda, \lambda')p_j(\lambda')$$

The total support given to label λ at node i is then the total neighborhood support and is hence equal to

$$s_i(\lambda) = \sum_{j=1}^{n} \sum_{\lambda'=1}^{m} r_{ij}(\lambda, \lambda')p_j(\lambda') \tag{2.10}$$

Let

$$\vec{p_i} = \begin{pmatrix} p_i(\lambda_1) \\ p_i(\lambda_2) \\ \\ p_i(\lambda_m) \end{pmatrix} \tag{2.11a}$$

and

$$\vec{s_i} = \begin{pmatrix} s_i(\lambda_1) \\ s_i(\lambda_2) \\ \vdots \\ s_i(\lambda_m) \end{pmatrix} \tag{2.11b}$$

A labelling $\vec{p_i}$ of the graph is consistent if and only if $\vec{p_i} = \vec{s_i}/(\vec{s_i}.\vec{1})$, with the vector $\vec{1}$ defined as $\vec{1} = (11 \quad 1)^T$ In other words, consistency is achieved only when the neighborhood support confirms the labelling assignment. The objective of relaxation labelling techniques is thus as follows: given an initial labelling assignment, iteratively

update this labelling until consistency is achieved. Thus, given the labelling assignment $\vec{p_i}^k$ at iteration k and the neighborhood support $\vec{s_i}^k$, consistency is achieved by minimizing

$$\| \vec{s_i}^k/(\vec{s_i}^k.\vec{1}) - \vec{p_i}^k \|$$

This is usually achieved by computing the projection $\vec{v_i}^k$ of the support vector on the hyperplane tangent to the constraint on the labelling assignment The new labelling assignment $\vec{p_i}^{k+1}$ is then equal to $\vec{p_i}^k + \vec{v_i}^k$ As far as line detection goes, the initial labelling assignments are derived by applying a number of orientation sensitive masks to the image The response of these coarse orientation estimators is then used in order to establish the initial labelling $\vec{p_i}^0$ The compatibilities between orientations at neighboring pixels on the other hand, are defined through a study of the differential geometry of lines and curves and encode the a priori contextual information relating to the problem domain[5] In an early work on this topic[6], eight orientation labels were assigned to each pixel, in addition to the "no-line" label The compatibilities between neighboring labels were defined as is shown in figure 2 4 Note also that the compatibilities used are chosen so as to enhance the "straightness" of a curve more than its 'curviness"

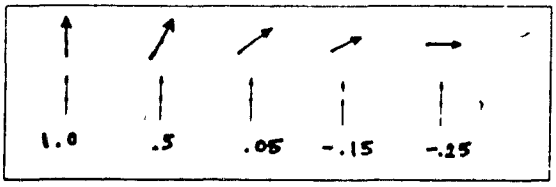


Figure 2.4 Orientation compatibilities

The result of applying this technique is generally very satisfactory. This is to be expected, since if the compatibility functions are properly chosen and if the initial labelling assignment is correct, the result will be a globally optimal assignment of orientation labels to the picture elements. The major drawback with this scheme however, resides in its excessive computational complexity As noted in [7], for a 256 × 256 size image with 8 labels per pixel, 524288 label certainties need be updated at every iteration This induces a

heavy computational burden on existing sequential machines and makes it impractical for a large number of applied vision tasks where time performance must be taken into account.

2.3 Search Techniques

Search techniques, also referred to as line tracking techniques, are another class of algorithms which use local constraints The simplest search technique is also the least efficient and consists of exhaustive search Given a starting point (usually identified as a bright point on a dark background), the objective is to select, among the immediate neighbors of this point, the highest intensity point, and from there on the process is repeated If the line has a consistently higher intensity than its background and if the background is relatively uniform and noise-free, then such a technique could work well Such is not the case however for real images, and false contour points are generated due to noise and the line to be tracked is sometimes artificially broken up In such cases, backtracking becomes necessary, and the set of possible contours grows exponentially large. To see the effect of noise on such an algorithm, assume that a line L is formed of N connected pixels $P_i(x_i, y_i)$ In the noise-free case, $f(P_i)$ is a maximum in a selected neighborhood of P_i (say the one fixed by the direction of search), and we have.

$$\begin{aligned} Pr(line \quad found) &= Pr(all \quad points \quad are \quad local \quad maxima) \\ &= \prod_{i=1}^{N} Pr(f(P_i) \quad local \quad maximum) \quad , \end{aligned} \tag{2.12}$$

In addition, assume

$$f(P_i) = \begin{cases} U > 0, & \text{if } P_i \in L: \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

We would like to evaluate the performance of this simple local maximum selector in the presence of noise For this purpose, we assume that a 2-D noise process η is added to the image function f We assume each sample $\eta(x, y)$ to be statistically independent of another sample $\eta(x', y')$ Without any loss of generality we assume $\eta(x, y)$ to be zero-mean with variance σ^2 Since the probability of finding a contour line is related to that of each contour point being a local intensity maximum in a selected neighborhood, we will concentrate on quantifying this latter probability

Assume a simplification of this general search problem, where the next pixel in the contour is to be found in a 4-connected neighborhood of the current pixel and where an aproximate a-priori knowledge of the direction of search reduces the problem to that of selecting the pixel having the highest intensity in a 2-pixel neighborhood (see figure 2 5). Thus

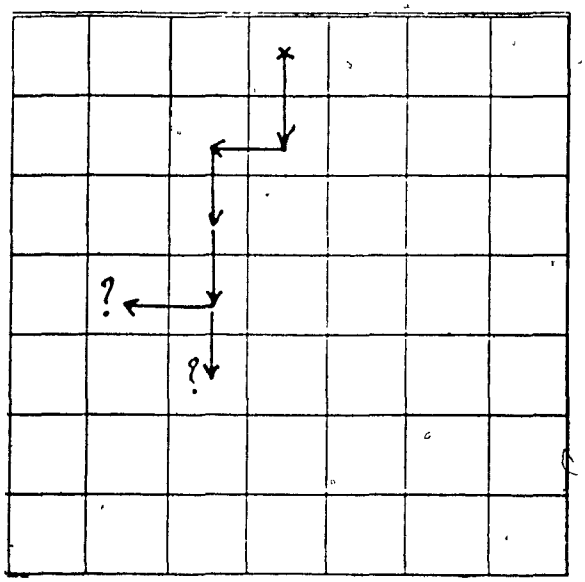


Figure 2.5 Contour line search

$$Pr(f(P_i) \quad local \quad maximum) \doteq Pr(f(P_i) > f(P_j)) \tag{2.14}$$

where P_i and P_j are the two possible candidate points Hence

$$Pr(f(P_i) \quad local \quad maximum) = Pr(f(P_i) - f(P_j) > 0) \tag{2.15}$$

Let $X_1 = f(P_i) - f(P_j)$ be the random variable corresponding to the difference in intensity of the two pixels. Also assume P_i is a contour point, while P_j is a background points Thus, P_i is distributed with mean U and variance σ^2, while P_j is zero-mean and with variance σ^2 Furthermore, since the samples are statistically independent, we obtain that X_1 is distributed with mean U and variance $2\sigma^2$ Thus,

$$Pr(f(P_i) \quad local \quad maximum) = Pr(X_1 > 0) \tag{2.16}$$

Using Chebyshev's inequality, we obtain:

$$Pr(|X_1 - U| < k\sqrt{2\sigma^2}) \geq 1 - 1/k^2 \quad k \in \mathbf{R}^+ \tag{2.17}$$

and hence

$$Pr(|X_1 - U| \geq k\sqrt{2\sigma^2}) \leq 1/k^2 \tag{2.18}$$

Thus, we have

$$Pr((-X_1 + U) \geq k\sqrt{2\sigma^2}) \leq 1/k^2 \tag{2.19}$$

and therefore

$$Pr(X_1 \leq (U - k\sqrt{2\sigma^2})) \leq 1/k^2 \tag{2.20}$$

From equation (2.20), it is clear that we are interested in the case where X_1 yields a positive value. We thus set $k = U/\sqrt{2\sigma^2}$, and we obtain

$$Pr(X_1 \leq 0) \leq 2\sigma^2/U^2 \tag{2.21}$$

and after some manipulation, we get:

$$Pr(X_1 > 0) \geq (1 - 2\sigma^2/U^2) \tag{2.22}$$

Returning to the previous equation, we then obtain:

$$Pr(f(P_i) \quad local \quad maximum) \geq (1 - 2\sigma^2/U^2) \tag{2.23}$$

and therefore, assuming $1 - 2\sigma^2/U^2 \geq 0$:

$$
\begin{aligned}
Pr(line\ \ found) &= Pr(all\ \ points\ \ are\ \ local\ \ maxima) \\
&= \prod_{i=1}^{N} Pr(f(P_i)\ \ local\ \ maximum) \\
&\geq \prod_{i=1}^{N} (1 - 2\sigma^2/U^2) \\
&\geq (1 - 2\sigma^2/U^2)^N
\end{aligned}
\tag{2.24}
$$

The expression $(1 - 2\sigma^2/U^2)^N$ is thus seen to constitute a lower bound on the probability that the contour line is correctly found We also note that for large values of the signal-to-noise ratio U/σ, this lower bound approaches 1 and hence the probability of correct line detection is increased What is interesting to note in this expression is the dependence of this probability on the signal-to-noise ratio U/σ and on the length N of the line the whole story of noise-sensitivity of contour tracking algorithms is embedded in this one inequality

An extension of this simple algorithm has been proposed by Montanari[8], which incorporates constraints other than connectedness More specifically, he introduces constraints on curvature in addition to constraints on intensity into a dynamic programming framework, where a global figure of merit related to both intensity and curvature is maximized The maximization of the criterion yields the lowest curvature curve with the highest contrast. Assuming the line to be formed of successively connected points P_1, P_2, P_n with $P_i(x_i, y_i)$, the global figure of merit is

$$
g(P_1, P_2, ..., P_n) = \sum_{i=1}^{n} f(P_i) - q \sum_{i=2}^{n-1} (d(P_{i+1}, P_i) - d(P_i, P_{i-1}) \mod 8) \tag{2.25}
$$

where f is the image intensity function, and $d(P_{i+1}, P_i)$ is the slope of the curve between points P_i and P_{i+1}.

The first term computes the overall intensity of the line, while the second computes its overall curvature. From the above, we see that the figure of merit can be expressed as a sum of figures of merit involving fewer variables and hence, owing to this separability,

the multistage decision process of dynamic programming can be used Although this technique possesses a global perspective which the previous one did not have, the computational time and storage requirements render it as impractical for any realistic application

Another algorithm which is based on local search is that of Shirai[9] Rather than using low level constraints such as curvature, however, it employs high level constraints given by a knowledge of the scene in which image lines are to be found As such therefore, its application is limited to images of polyhedra and works as follows Given a set of contour lines initially found, compute the position of the vertices and from these hypothesize possible line arrangements These line arrangements are given by a study of visible edges in polyhedral scenes and could be due to internal or external boundaries and to junctions of faces or occlusions The verification stage is done by computing an error or dispersion measure between the hypothesized line and the feature points in the image which are supposed to support it Although this algorithm has proven to be successful in finding lines in images of polyhedra, its use of high level knowledge constrains its applicability to only that class of images and hence this algorithm is not suitable for general purpose line detection in a bottom-up context

2.4 Burns et al.'s Technique[10]

This algorithm explicitely uses the gradient orientation information as well as the gradient magnitude and hence is the closest, among the algorithms mentioned, to the one presented in this thesis. The algorithm can be summarized in four steps in a first step, pixels are grouped into line support regions based on similarity of gradient orientation (see figure 2 6).

In a second step, the intensities of the pixels in this region are approximated by a planar surface In other words, the edge itself is approximated by a plane. Then, information related to the line characteristics (such as contrast or width) is extracted from

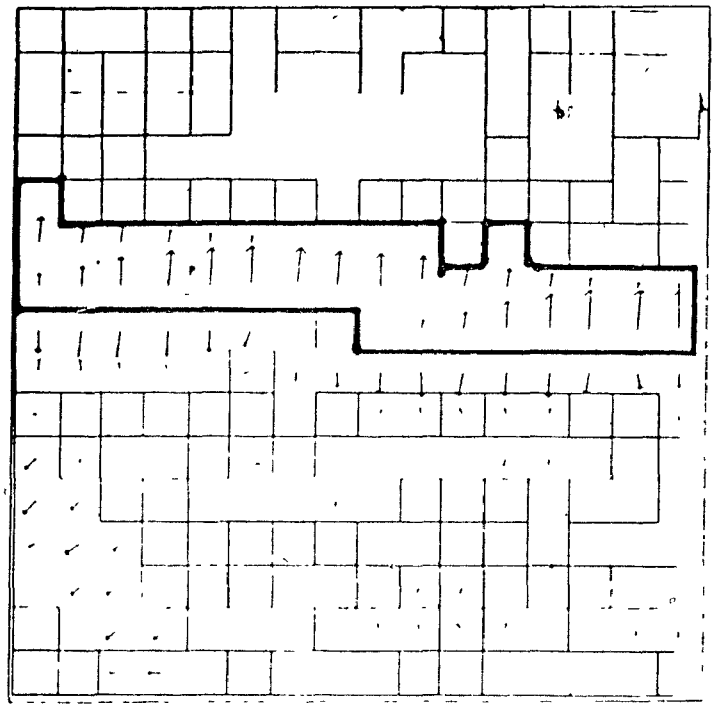


Figure 2.6 Line support region (from [10])

the planar fit and from the line support region Once these attributes are extracted, the lines can be symbolically represented and hence, selected, based on the desire to isolate certain image events. In order to evaluate the merits and weaknesses of this algorithm, each of these steps is described in more detail

The first step in the algorithm is pixel grouping, based on gradient orientation Thus, a specific operator must be used in order to allow the extraction of gradient orientation Too little neighborhood support for the operator will increase noise sensitivity, while too large a support will average out too much detail The primary requirement that is imposed is that no fine detail be lost at the operator convolution stage Thus, the smallest possible mask, which is a 2×2 pixel operator, is used Once the orientations at each pixel are estimated, the pixels must be grouped based on orientation similarity Region growing could be a possible solution, but due to its local nature, it could cause disastrous results since intermediate orientations are usually also created at discontinuities Instead, therefore, the orientations are quantized into a number of distinct values, and pixels are assigned a label which corresponds to the value of this quantization Then pixels having the

same label are grouped together via a connected component labelling algorithm In effect, the number of allowed quantization values determines how much variation in orientation is allowed within a line support region The finer this quantization, the smaller the tolerance The problem with this approach is that lines are sometimes artificially broken and conversely, sometimes overmerged To overcome this problem, different quantizations are used and the one which is kept is the one which yields the longest line Another problem with this approach is that whenever grouping problems occur, the shape of the line support regions is dramatically changed

Once the line support region is found, a weighted least-squares fit of a planar surface is made to the intensities of the pixels forming the region The weights are determined by the gradient magnitudes at those pixels The associated fit parameters can then be used to describe the type of line detected Additional attributes can also be extracted from the line support region, such as iso-contours (contours of pixels of equal intensity), orientation variance, piecewise average orientation, which all give a measure of the straightness of the line, and contrast, width and steepness, which give a measure of the strength of the line as well as its spatial spread

This algorithm, as reported in [10], has been successfully applied to images of outdoor scenes and aerial pictures Of all the algorithms that have been reviewed, Burns et al.'s performs the best as far as the final result and the associated computational complexity are concerned This is due to the fact that they use gradient orientation information, in addition to gradient magnitude information With this scheme, however, placement of lines can be skewed under slow intensity changes Also, the line support region computation is based on a coarse orientation quantization which could cause artificial breakups in the line segments

This thesis presents an algorithm, originally reported in a report in 1985 (see [11]), which overcomes the above mentioned problems, while yielding equivalent performance.

Chapter 3 Line Detection: a Prediction/Verification Approach

3.1 Image Pre-processing

Before detecting lines, we shall first define exactly what we mean by straight line segments In images, we can basically distinguish between two types of linear structures: the first type is a linear arrangement of white pixels on a black background or vice-versa. This corresponds to the case where the object perceived itself has a linear structure, such as blood vessels in biomedical images or roads in aerial maps[12] The second type of linear structure corresponds to linear contour lines. This corresponds to the case where the object perceived contains linear or planar boundaries (e g polyhedra) and object/background or object/object occlusions give rise to perceived linear contours Although as will be seen later, the difference between detecting these two different structures is basically a matter of preprocessing and does not change the essence of the algorithm presented in this thesis, we will concentrate mostly on detecting contour lines since we believe that they arise more frequently in the applied vision context where the detection of lines is not an end in itself, but a means towards the detection of higher level structures.

Two main trends emerge from the early processing of images The first is the contour based approach, where object contours, also called "edges", are to be found in order to help the recognition stage. The second type of low level processing is the region based

approach, where the emphasis is not put on finding object contours (formed by occlusions), but rather on finding object regions, i e. regions in the image which correspond to distinct objects. Since our objective is to find straight contour lines, we will adopt the former approach and we will use schemes that will enable us to find the set of those picture elements which constitute object boundaries In order to do this, we will make use of edge enhancement operators and attempt to quantify their performance

3.1.1 Edge Enhancement

A number of edge enhancing operators have been developed to highlight region boundaries[13][14] Consider a continuous image function f defined on a continuous domain and analytic at-every point In such an image object boundaries would correspond to points in the image where the intensity function changes rapidly This rapid change could be quantified by considering the spatial derivative of the function Thus, object boundaries would correspond to points in the image where the derivative of the function is "large " How large is large is the legitimate question that arises at this point, and, in many cases, it is not possible to distinguish between a true object boundary (an edge) and points on the image which correspond to noisy variations, using a simple differential operator as an edge detector Other techniques that have been developed assume different edge models and hence different criteria are used for edge detection[15] Although the performance of the line detector proposed in this thesis depends to a large extent on the performance of the edge enhancement operator used, we shall not focus on the details of this operation Rather, the most suitable edge operator among the ones most used will be found, according to a criterion given below These consist of mere approximations to the differential operator and are hence termed "gradient operators" The operators that we will consider are the Roberts cross-operator[16], the Sobel operator[17], and the Prewitt operator[18]

3.1.1.1 The Roberts Cross-operator

The Roberts cross-operator consists of two orthogonal 2×2 pixel masks (see

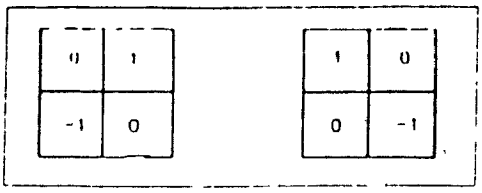


Figure 3.1 The Roberts operator

figure 3.1).

Assuming an image function f, the response e_x of the first mask is given by:

$$e_x(i,j) = f(i,j) - f(i+1,j+1) \tag{3.1a}$$

and the response e_y of the second mask is given by:

$$e_y(i,j) = f(i+1,j) - f(i,j+1) \tag{3.1b}$$

The magnitude of the edge at the point (x,y) is then given by any choice of norm (L_1, L_2, ... L_∞) Usually however, the Euclidian (L_2) norm is used and the edge magnitude equals

$$e(i,j) = \sqrt{e_x(i,j)^2 + e_y(i,j)^2} \tag{3.2}$$

Also, the orientation $\theta(i,j)$ of the gradient vector with respect to the $x-$ axis is given by

$$\theta(i,j) = tan^{-1}(e_x/e_y) + \pi/4 \tag{3.3}$$

We are interested in evaluating the noise performance of this operator Clearly, this performance will depend on the performance of each of the masks Consider a two dimensional point noise process η with $\eta(i,j)$ being independent zero-mean identically distributed random variables with variance σ^2 We are interested in the variance of e_x and e_y in response to the noise process We then obtain

$$e_x(i,j) = \eta(i,j) - \eta(i+1,j+1) \tag{3.4a}$$

and

$$e_y(i,j) = \eta(i+1,j) - \eta(i,j+1) \tag{3.4b}$$

hence:

$$Var_x = Var(e_x(i,j)) = Var(\eta(i,j)) + Var(\eta(i+1,j+1)) = 2\sigma^2 \qquad (3.5a)$$

$$Var_y = Var(e_y(i,j)) = Var(\eta(i+1,j)) + Var(\eta(i,j+1)) = 2\sigma^2 \qquad (3.5b)$$

We thus obtain:

$$Var_x/\sigma^2 = Var_y/\sigma^2 = 2\sigma^2/\sigma^2 = 2 \qquad (3.6)$$

The output noise variance is therefore twice as large as the input noise variance. With this operator, therefore, the noise is not attenuated but rather is amplified and hence, this operator is very noise sensitive.

3.1.1.2 The Sobel Operator

The Sobel operator consists of two orthogonal 3 · 3 pixel masks (see figure 3.2).

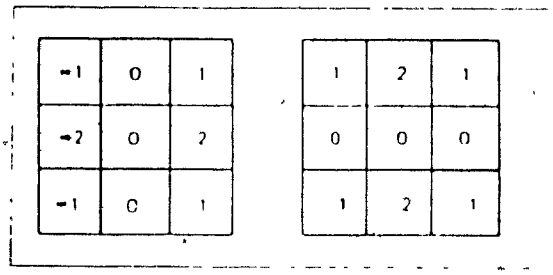


Figure 3.2 The Sobel operator

Assuming an image function f, the response e_x of the first mask is given by:

$$\begin{aligned} e_x(i,j) = &(f(i+1,j+1) - f(i-1,j+1))/4 + \\ &(f(i+1,j) - f(i-1,j))/2 + \\ &(f(i+1,j-1) - f(i-1,j-1))/4 \end{aligned} \qquad (3.7a)$$

and the response e_y of the second mask is given by

$$\begin{aligned} e_y(i,j) = &(f(i-1,j+1) - f(i-1,j-1))/4 + \\ &(f(i,j+1) - f(i,j-1))/2 + \\ &(f(i+1,j+1) - f(i+1,j-1))/4 \end{aligned} \qquad (3.7b)$$

Again, assuming an input noise process η with the same properties as before, we obtain the following output variances:

$$Var_x = Var(e_x(i,j)) = 3\sigma^2/4 \tag{3.8a}$$

and similarly,

$$Var_y = Var(e_y(i,j)) = 3\sigma^2/4 \tag{3.8b}$$

The ratio of the output to the input noise variances is then

$$Var_x/Var(\eta(i,j)) = Var_y/Var(\eta(i,j)) = 3\sigma^2/4\sigma^2 = 3/4 \tag{3.9}$$

In this case, the output noise variance is smaller than the input noise variance and hence, the noise is to some extent reduced. This is due to the fact that the operation performed in the Sobel operator is not merely a straight differencing of intensities at two different pixel locations, but rather a differencing of their neighborhood averages. It is this averaging that reduces the noise variance. This stands in direct contrast with the Roberts operator which does not perform any kind of pre-averaging.

3.1.1.3 The Prewitt Operator

The Prewitt operator also consists of two orthogonal 3 × 3 pixel masks (see figure 3.3).

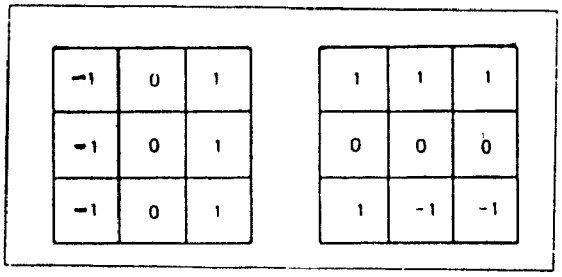


Figure 3.3 The Prewitt operator

Given an image function f, the response e_x of the first mask is given by:

$$\begin{aligned} e_x(i,j) =& (f(i+1,j+1) - f(i-1,j+1))/3+ \\ & (f(i+1,j) - f(i-1,j))/3+ \\ & (f(i+1,j-1) - f(i-1,j-1))/3 \end{aligned} \tag{3.10a}$$

and the response e_y of the second mask is given by

$$\begin{aligned} e_y(i,j) =& (f(i-1,j+1) - f(i-1,j-1))/3+ \\ & (f(i,j+1) - f(i,j-1))/3+ \\ & (f(i+1,j+1) - f(i+1,j-1))/3 \end{aligned} \tag{3.10b}$$

Again, assuming an input noise process η with the same properties as before, we obtain the following output variances:

$$Var_x = Var(e_x(i,j)) = 2\sigma^2/3 \tag{3.11a}$$

and similarly,

$$Var_y = Var(e_y(i,j)) = 2\sigma^2/3 \tag{3.11b}$$

The ratio of the output to the input noise variances is then

$$Var_x/Var(\eta(i,j)) = Var_y/Var(\eta(i,j)) = 2/3 \tag{3.12}$$

In this case, noise attenuation is even stronger than with the Sobel operator. This is due to the fact that the averaging that is performed before the differencing operation takes place is not a weighted averaging, but rather, all the samples are given equal weight.

3.1.1.4 Choice of Local Operator

Of the three operators that have been presented, only the Sobel operator and the Prewitt operator attenuate noise. We thus discard the Roberts cross-operator because of its low noise immunity. Using noise performance as the only selection criterion, one would choose the Prewitt operator. Noise performance is not everything, however. As important as noise performance is the ability of the operator to properly localize the edge. With the

Prewitt operator, the averaging operations that are performed attribute equal weight to all intensity samples, and hence, any permutations of those intensity samples will not change the operator response,while changing the structure of the edge to be detected. With the Sobel operator, however, closer samples are attributed larger weight and hence the spatial structure of the edge to be detected is more accurately captured. In addition, given that the noise performance of the Sobel operator is comparable to that of the Prewitt operator, the Sobel operator will be used in order to enhance edges in intensity images Once the edges are enhanced, however, only those that correspond to the desired image events must be kept. Achieving this is not possible unless some a priori knowledge is available about the scene whose image is being processed This a priori knowledge usually concerns the gradient magnitude and is based on the premise that strong edges correspond to desirable image features while weak edges are induced by noisy variations This assumption breaks down for images with strong texture patterns (e g. outdoor scenes) but is a reasonable one for most industrial vision tasks where most objects have uniform intensities and the contrasts are strong. Hence, we shall use this assumption in order to "detect" edges after having enhanced them

3.1.2 Edge Detection

The result of edge detection is a binary image: either a pixel is an edge element or it is not. In order to achieve this result, the edge magnitude $e(x,y)$ of a point $P(x,y)$ is computed, using the two orthogonal edge components $e_x(x,y)$ and $e_y(x,y)$ and an arbitrary choice of norm (L_1, L_2, . . . L_∞) The point $P(x,y)$ is considered to be an edge element if and only if $e(x,y) > T$ where T is an arbitrary positive constant. Otherwise, the point $P(x,y)$ is considered to be a background point. The intensity value T, which delimits the intensity values comprising edge and non-edge pixels is usually referred to as the "intensity threshold," and this operation as "intensity thresholding." From the above definition, it can be seen that thresholding is a global operation. Points in the image are assigned different labels (e.g edge and non-edge) based only on their intensity and independently of the

local properties they might have. In order to quantify the performance of the thresholding operation, a criterion which reflects how good or how bad the final result is has to be defined. For this purpose, we define a probability of error for the edge detection stage as:

$$\begin{aligned} Pr(error) = & Pr(edge \quad classified \quad as \quad non-edge) + \\ & Pr(non-edge \quad classified \quad as \quad edge) \end{aligned} \tag{3.13}$$

Let R_E be the set of all edge pixels and R_N be the set of all non-edge pixels The union $R_E \cup R_N$ of these two sets is the set of all images points In addition, the intersection $R_E \cap R_N$ of these sets is the null set since no pixel can assume both states The probability of occurence P_E of an edge pixel is defined as

$$P_E = Card(R_E) / Card(R_E \cup R_N) \tag{3 14}$$

and is the ratio of the number of edge points to the total number of image points $Card$ refers to the cardinality of the set in question Similarly, the probability of occurence P_N of a non-edge pixel is defined as

$$P_N = Card(R_N) / Card(R_E \cup R_N) \tag{3.15}$$

With these definitions, the probability of error is then rewritten as

$$\begin{aligned} Pr(error) = & Pr(e(x,y) > T / P(x,y) \in R_N) P_N + \\ & Pr(e(x,y) < T / P(x,y) \in R_E) P_E \end{aligned} \tag{3.16}$$

We shall now study the conditional probabilities which contribute to the classification error. Assume the set R_E has an intensity distribution with mean μ_E and variance σ_E^2, while the set R_N of non-edge pixels has an intensity distribution with mean μ_N and variance σ_N^2

Assuming $\mu_n < T < \mu_E$, we have

$$Pr(e(x,y) > T / P(x,y) \in R_N) \leq 1 - Pr(|e(x,y) - \mu_N| < (T - \mu_N)) \tag{3.17}$$

and

$$Pr(e(x,y) < T / P(x,y) \in R_E) \leq 1 - Pr(|e(x,y) - \mu_E| < (\mu_E - T)) \tag{3.18}$$

Thus:

$$Pr(error) \leq P_N(1 - Pr(|e(x,y) - \mu_N| < (T - \mu_N))) + P_E(1 - Pr(|e(x,y) - \mu_E| \leq (\mu_E - T))) \quad (3.19)$$

Let $T = \mu_N + k_1\sigma_N$. We then obtain $k_1 = (T - \mu_N)/\sigma_N$, and from Chebyshev's theorem:

$$Pr(|e(x,y) - \mu_N| < (T - \mu_N)) \geq 1 - \sigma_N^2/(T - \mu_N)^2 \quad (3.20)$$

Now, let $T = \mu_E - k_2\sigma_E$. We then get $k_2 = (\mu_E - T)/\sigma_E$, and from Chebyshev's theorem:

$$Pr(|e(x,y) - \mu_E| < (\mu_E - T)) \geq 1 - \sigma_E^2/(T - \mu_E)^2 \quad (3.21)$$

After some simple manipulations, we obtain

$$Pr(error) \leq P_N\sigma_N^2/(T - \mu_N)^2 + P_E\sigma_E^2/(\mu_E - T)^2 \quad (3.22)$$

Let

$$UB(T) = P_N\sigma_N^2/(T - \mu_N)^2 + P_E\sigma_E^2/(\mu_E - T)^2 \quad (3.23)$$

be the upper bound on the error probability. The optimal threshold T^* is the one which minimizes $UB(T)$. To find it, we let $dUB(T)/dT = 0$. After simple manipulations, the optimal threshold is found to be

$$T^* = (\mu_E(P_N\sigma_N^2)^{1/3} + \mu_N(P_E\sigma_E^2)^{1/3})/((P_N\sigma_N^2)^{1/3} + (P_E\sigma_E^2)^{1/3}) \quad (3.24)$$

It can be seen that if the two sets have similar size and variances, the optimal threshold will be halfway between the two means, which is as expected. The minimum value UB^* of $UB(T)$ associated with the optimal threshold T^* is itself found to be.

$$UB^* = UB(T^*) = ((P_N\sigma_N^2)^{1/3} + (P_E\sigma_E^2)^{1/3})^3/(\mu_E - \mu_N)^2 \quad (3.25)$$

The above equality shows the minimum value that can be attained with the upper bound on the probability of misclassification. As expected, the larger the variances of the two classes (i.e. edge and non-edge) and the smaller their mean difference, the larger the upper bound on the error probability. Conversely, the smaller the variances and the larger the mean difference, then the smaller the upper bound will be. It is now understood

why the aim has been, when choosing the local operators, to reduce their output noise variance. In principle, the above derivations give us the necessary tools for selecting the optimal threshold However, this requires a priori knowledge concerning the probability of occurence of each class in addition to some statistical properties which are not usually available beforehand This is why, in practice, other criteria are used for threshold selection and which do not depend on a priori quantified knowledge [19]

3.2 The Algorithm

3.2.1 Basic Idea

The algorithm proposed in this thesis is based on the following idea, use as much local information as possible in order to have accuracy in the line detection process, and use as much global information as possible in order to ensure noise immunity This idea illustrates the basic tradeoff inherent in all line detection schemes In order to satisfy this tradeoff, a two-step procedure is used In the first step, local information is used in order to predict the line or lines that could exist, while in the second step, these predictions are verified against more global information This paradigm is also known as the hypothesis prediction/verification paradigm[20][21], which can be stated in the following informal manner for a given problem, hypothesize all possible solutions to this problem, reject those solutions that do not obey the given constraints, and retain only those that solve the problem under all prescribed constraints The prediction/verification paradigm is thus seen to consist of two parts: a predictor, which enumerates all possible solutions, and a verifier, which evaluates each proposed solution, either accepting it or rejecting it (see Figure 3.4)

Stated in a more formal manner, the predictor generates a set H of N hypotheses $\{h_1, h_2, ..., h_N\}$. The verifier then maps the set H into a subset H' such that

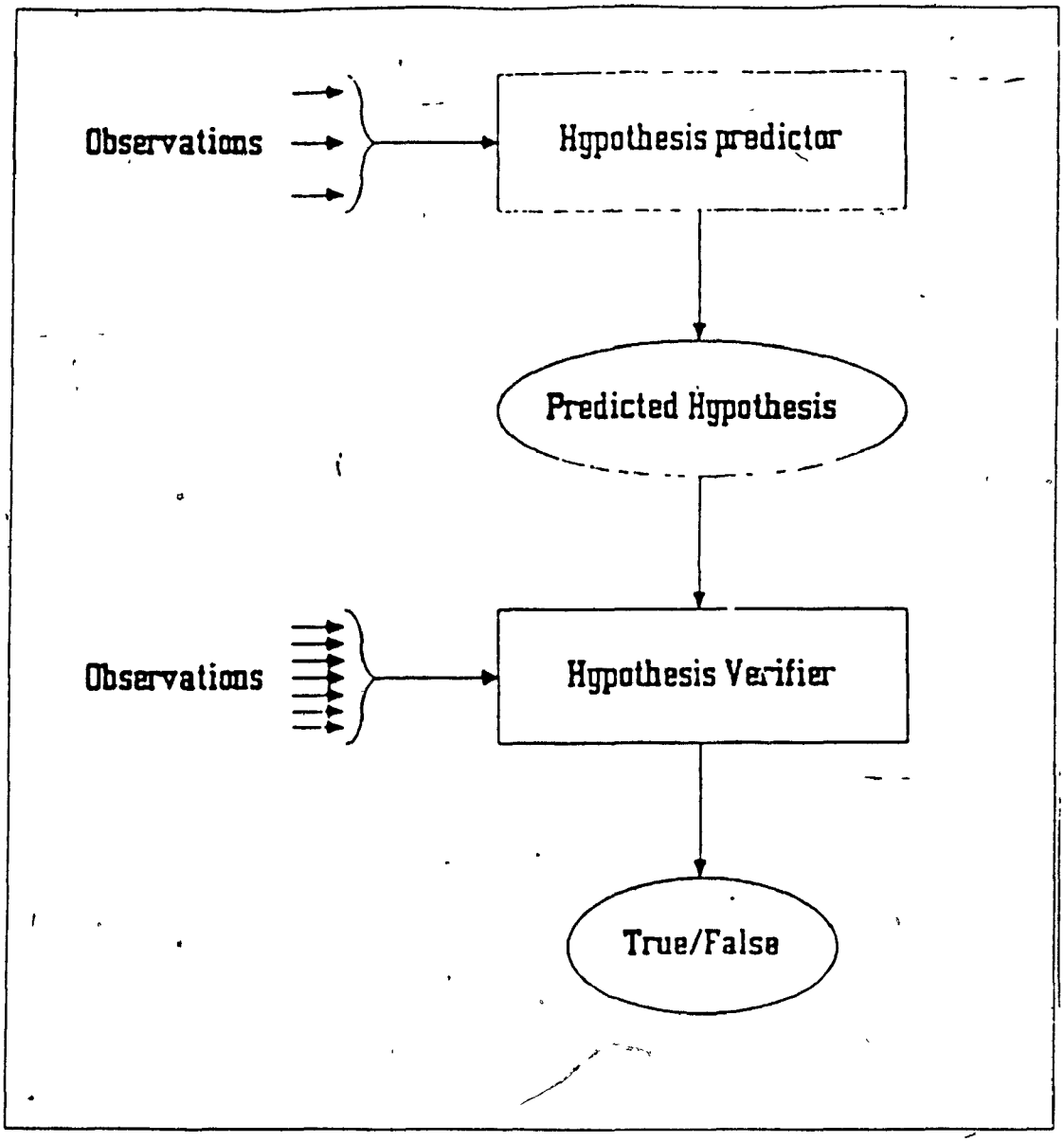


Figure 3.4 The hypothesis prediction/verification paradigm

all hypotheses $h_i \in H^T$ evaluate to true The truth value of the hypotheses is evaluated based on criteria that are developed later on Good predictors have the following two properties: they are complete, in the sense that they produce all possible hypotheses Also, they are informed in the sense that they use possibility-limiting information, restricting the hypotheses they propose accordingly Informability is important, otherwise, hypothesis verification becomes an exhaustive tree-search algorithm The prediction of hypotheses, therefore, must be based on strong heuristics

The AI hypothesis prediction/verification paradigm is very close to the concept of Kalman Filtering often encountered in estimation theory The underlying formulation of the Kalman Filter is the following

Estimation = Prediction + Correction

In this model, the estimation of a state at some time index n is equalized to a prediction of this same state based on time samples preceeding or equal to n 1, added to a correction factor that is a function of the predicted and observed states The correction factor is in some sense the 'prediction verifier " Were the prediction exact, the correction term would be zero In our model, the prediction is based on strictly local information In other words, a given straight line segment is predicted to exist, based only on the features at one pixel location The verification however, is based on a more global analysis, since statistical tests are now performed on a neighborhood of this pixel location

3.2.2 Hypothesis Prediction

Consider a picture element at (x_c, y_c) To this pixel, we associate, through gradient filtering, the two orthogonal edge components (e_{x_c}, e_{y_c}) Assuming (x_i, y_i) to be a point on a line segment S, the equation

$$(x_i - x_c)e_{x_c} + (y_i \quad y_c)e_{y_c} = 0 \tag{3 26}$$

has to be met by all points (x_i, y_i) that are located on the line segment S Equation (3) is merely a mathematical representation of the statement that edge vectors are orthogonal to region boundaries. Let us assume that we wish to detect all straight line segments that are composed of $2n + 1$ picture elements (i e are of order n) The hypothesis predictor then predicts the following hypothesis which we call H_0

There is a segment S of order n centered at (x_c, y_c) such that

$$\forall (x_i, y_i) \in S \cdot (x_i - x_c)e_{x_c} + (y_i - y_c)e_{y_c} = 0 \tag{3:27}$$

The set of hypotheses is thus formed from the single hypothesis H_0.

The verifier now must evaluate the predicted hypothesis. This verification is done using statistical tests performed on the predicted line segment S Let

$$E_x = \{\frac{e_{x_i}}{N_i} \in \mathcal{R} / (x_i, y_i) \in S\} \tag{3.28a}$$

$$E_y = \{\frac{e_{y_i}}{N_i} \in \mathcal{R} \cdot (x_i, y_i) \in S\} \tag{3.28b}$$

with

$$N_i = \sqrt{e_{x_i}^2 + e_{y_i}^2} \tag{3.29}$$

E_x and E_y are respectively the horizontal and vertical normalized edge components of the picture elements that are located on the predicted line segment S The edge components have been normalized in order to provide robustness to changes in intensity In other words, we are now dealing with unit vectors and the only information we are interested in is their orientation.

3.2.3 Hypothesis Verification

We assume a statistical model for the purpose of our analysis. We interpret E_x as a set of $2n+1$ observations on a random variable X normally distributed with mean μ_x and variance σ_x^2. Similarly, we interpret E_y as a set of $2n+1$ observations on a random variable Y normally distributed with mean μ_y and variance σ_y^2. In other words, we interpret E_x (and E_y) as a sample set drawn from a population having an infinity of elements This infinitely large population would be the continuous line segment; since the image is defined over a discrete two-dimensional domain however, only a finite number of points on this continuous segment can be observed, and these points correspond to the discrete values at which the image function is defined

In the ideal case, i.e. the case where there is a line segment of order n centered at (x_c, y_c), the mean of the random variable X has to be equal to e_{x_c}/N_c and the mean

of the random variable Y to e_{y_c}/N_c In addition, the variances of the random variables X and Y would be zero. This is usually not the case! Small variances of X and Y however, indicate a uniformity in the orientation of the edge elements in the population In other words, if the variance in the distribution of the $x-$ and $y-$ edge components is small, there is a large probability that the predicted line segment does actually exist If on the other hand, the variances are large, the probability that the predicted line exists is fairly small since this would mean that the edge elements located on the predicted line segment have random orientations These observations are essential to the algorithm, and hypothesis verification is based on these very same observations of the behavior of the mean and the variance in the normalized edge components

It should be noted that the means and variances that have been mentioned so far concern the population and not the sample set The only statistical tests that can be performed however relate to the sample set Thus, we must investigate the significance of the statistical measurements that are performed on the sample set, with respect to the overall population This significance test is performed on the sample mean using Student's t-distribution and on the sample variance using chi-square (χ^2) tests We define the sample variances S_X^2 and S_Y^2 of X and Y, respectively, as

$$S_X^2 = \frac{1}{2n} \sum_{i=1}^{2n+1} (e_{x_i} N_i - \overline{S_X})^2 \tag{3.30a}$$

and,

$$S_Y^2 = \frac{1}{2n} \sum_{i=1}^{2n+1} (e_{y_i}/N_i - \overline{S_Y})^2. \tag{3.30b}$$

also $\overline{S_X}$ and $\overline{S_Y}$ are the sample means of X and Y, respectively, defined as

$$\overline{S_X} = \frac{1}{2n+1} \sum_{i=1}^{2n+1} e_{x_i}/N_i \tag{3.31a}$$

$$\overline{S_Y} = \frac{1}{2n+1} \sum_{i=1}^{2n+1} e_{y_i}/N_i \tag{3.31b}$$

Note that in our definition of the variance, the summation has been divided by $2n$ and not by $2n+1$, and so, the sample variance is an unbiased estimator for the true variance of

the population. The sample mean, as defined in equations (3.31a) and (3.31b) is also an unbiased estimator for the true mean of the population

Since the random variables X and Y are assumed to be normally distributed, the statistics

$$\frac{\overline{S_X} - \mu_x}{\sqrt{S_X^2/(2n+1)}}, \quad \frac{\overline{S_Y} - \mu_y}{\sqrt{S_Y^2/(2n+1)}}$$

are t- distributed (Student's t distribution) with $2n$ degrees of freedom, and the statistics

$$\frac{2nS_X^2}{\sigma_x^2}, \quad \frac{2nS_Y^2}{\sigma_y^2}$$

are chi-squared with $2n$ degrees of freedom The reduction in the number of degrees of freedom from $2n+1$ to $2n$ is due to the fact that the population means and variances are not known and can be only estimated by sample measurements

The main objective underlining the above expressions is the ability to generate confidence intervals on the population statistics[22] In other words, we desire to establish lower and upper bounds on the population mean and variance, with some confidence, based on the mean and the variance of the sample set Once bounds on the population statistics have been estimated, there is enough evidence to accept or reject the predicted hypothesis concerning the existence of a line segment

The $100(1-\alpha)\%$ confidence intervals on the population means μ_x and μ_y (with $0 \leq \alpha \leq 1$) are respectively defined as

$$M_X = (\overline{S}_X - t_{\alpha/2}\sqrt{S_X^2/(2n+1)}, \overline{S}_X + t_{\alpha/2}\sqrt{S_X^2/(2n+1)}) \qquad (3.32a)$$

and

$$M_Y = (\overline{S}_Y - t_{\alpha/2}\sqrt{S_Y^2/(2n+1)}, \overline{S}_Y + t_{\alpha/2}\sqrt{S_Y^2/(2n+1)}) \qquad (3.32b)$$

The $100(1-\alpha)\%$ confidence intervals on the population variances σ_x^2 and σ_y^2 (with $0 \leq \alpha \leq 1$) are respectively defined as

$$V_X = (\frac{2nS_X^2}{\chi^2_{\alpha/2}}, \frac{2nS_X^2}{\chi^2_{1-(\alpha/2)}}) \qquad (3.33a)$$

and

$$V_Y = \left(\frac{2nS_Y^2}{\chi^2_{\alpha/2}}, \frac{2nS_Y^2}{\chi^2_{1-(\alpha/2)}}\right) \tag{3.33b}$$

Now that the bounds on the population mean and variance have been established, a procedure must be defined upon which the prediction verifier can base itself in order to infer whether or not a predicted line segment does actually exist As was mentioned previously, if the variance of the orientations is small and if the mean of the orientations is close to the orientation of the predicted line segment then the probability of the line segment prediction being true is quite large As far as the population variances σ_x^2 and σ_y^2 are concerned therefore, we define a simple threshold on the upper bound of the confidence intervals V_X and V_Y Note that the assumption of normality of the random variables X and Y is not independent of the sample variances and that this assumption is correct only for small sample variances As for the population means μ_x and μ_y, we are interested in knowing how close the mean orientation of the edge elements on the predicted line segment is to the orientation of the predicted line segment, i e how close μ_x and μ_y are to the normalized x- and y- components of the edge element at which the line segment is centered Since the population means belong to the intervals M_X and M_Y (with a degree of confidence of $(1 - \alpha)$), we use the bounds on these confidence intervals in order to evaluate the discrepancy between the population means and the normalized x- and y- components of the edge element at which the predicted line segment is centered If this discrepancy is large, the probability of all edge elements on the predicted line segment having the same orientation as the line segment itself is quite small Let D_X be the largest difference between e_{x_c}/N_c and the bounds of the interval M_X, and D_Y the largest difference between e_{y_c}/N_c and the bounds of the interval M_Y, N_c being equal to $\sqrt{e_{x_c}^2 + e_{y_c}^2}$ In other words,

$$D_X = MAX(\,|e_{x_c}/N_c - \bar{S}_X - t_{\alpha/2}\sqrt{S_X^2/(2n+1)}|,\ |e_{x_c}/N_c - \bar{S}_X + t_{\alpha/2}\sqrt{S_X^2/(2n+1)}|)$$

and

$$D_Y = MAX(\,|e_{y_c}/N_c - \bar{S}_Y - t_{\alpha/2}\sqrt{S_Y^2/(2n+1)}|,\ |e_{y_c}/N_c - \bar{S}_Y + t_{\alpha/2}\sqrt{S_Y^2/(2n+1)}|)$$

Summarizing the hypothesis verification procedure, we have

Accept H_0 if and only if:

$$\frac{2nS_X^2}{\chi^2_{1-(\alpha/2)}} \leq \sigma_T^2 \text{ AND } \frac{2nS_Y^2}{\chi^2_{1-(\alpha/2)}} \leq \sigma_T^2 \text{ AND } D_X \leq \Delta_\mu \text{ AND } D_Y \leq \Delta_\mu$$

Reject H_0 if and only if:

$$\frac{2nS_X^2}{\chi^2_{1-(\alpha/2)}} > \sigma_T^2 \text{ OR } \frac{2nS_Y^2}{\chi^2_{1-(\alpha/2)}} > \sigma_T^2 \text{ OR } D_X > \Delta_\mu \text{ OR } D_Y > \Delta_\mu$$

where σ_T^2 is defined as the threshold variance and Δ_μ as the threshold mean difference A complete description of the algorithm is shown in Figure 3 5 For the sake of completeness, this algorithm is summarized as follows: in a first step, the intensity edges in the image are enhanced This is done using the Sobel gradient operator, although other operators with similar or superior performance could be used as well In a second step, the intensity edges are thresholded, and as a result, a binary image formed of edge elements and non-edge elements is obtained This binary image is then scanned in a sequential manner, and each edge element is in turn and independently considered At each of these edge elements, a line of specified length is predicted, which orientation is given by the gradient orientation at the edge element considered Statistical tests on the gradient orientation of the samples that constitute this predicted line are then performed and if the orientations of the samples are similar to the orientation of the predicted line segment, the latter is accepted, otherwise, it is rejected

3.2.4 Statistical Model Revisited

The main assumption that was taken in the preceeding section that justified the use of interval estimation was the fact that the normalized edge components were normally distributed. Clearly, this assumption breaks down when the predicted line segment spans a variety of different image structures, since, in this case, the samples would not originate from one unique population In this case, the variance of the distribution of the normalized edge elements would be large, depending on the uniformity of the underlying samples.

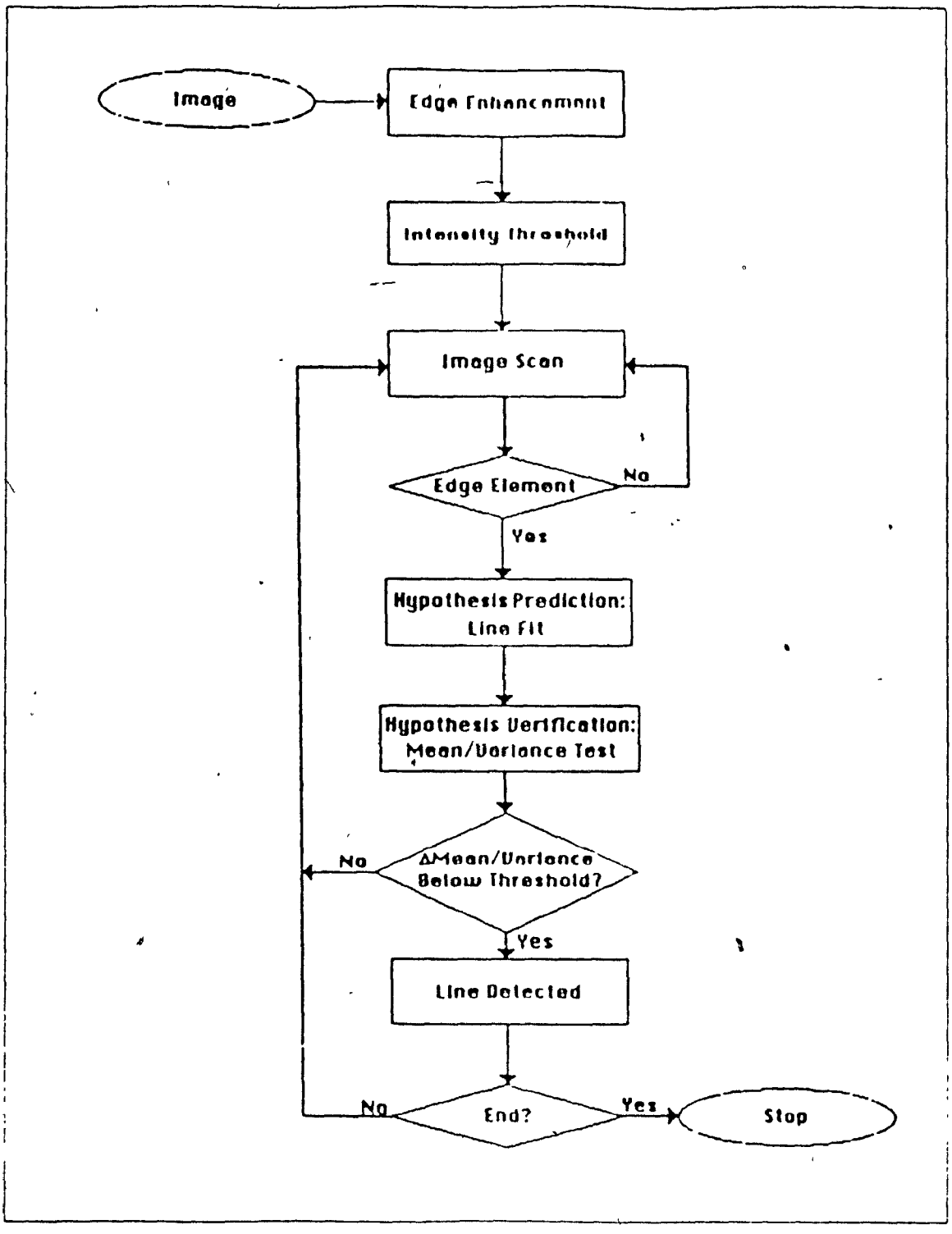


Figure 3.5 The algorithm

When the variance in the distribution is small, however, the samples tend to cluster around one mean, and here the normality asssumption holds Since one cannot always assume

a normal distribution, it is possible to use only sample measurements in the hypothesis verification process. Defining the sample deviation as

$$d_{2X} = \sqrt{\frac{1}{2n+1} \sum_{i=1}^{2n+1} (e_{x_i}/N_i - e_{x_c}/N_c)^2} \tag{3.34a}$$

and

$$d_{2Y} = \sqrt{\frac{1}{2n+1} \sum_{i=1}^{2n+1} (e_{y_i}/N_i - e_{y_c}/N_c)^2} \tag{3.34b}$$

we get the dispersion of the edge elements that are located on the predicted line segment with respect to the parameters of this same line segment The above dispersion measure is only a variation of one special distance measure, also called Euclidian distance In order to quantify this dispersion, one could also use a variant of the L_1 norm, defined as

$$d_{1X} = \frac{1}{2n+1} \sum_{i=1}^{2n+1} |e_{x_i}/N_i - e_{x_c}/N_c| \tag{3.35a}$$

and

$$d_{1Y} = \frac{1}{2n+1} \sum_{i=1}^{2n+1} |e_{y_i}/N_i - e_{y_c}/N_c| \tag{3.35b}$$

or the L_∞ norm, defined as:

$$d_{\infty X} = \max_i |e_{x_i}/N_i - e_{x_c}/N_c| \tag{3.36a}$$

and

$$d_{\infty Y} = \max_i |e_{y_i}/N_i - e_{y_c}/N_c| \tag{3.36b}$$

The difference between these different norms resides in the following: the L_1 norm provides a lot of averaging and hence, the final dispersion measure is relatively insensitive to large deviations in isolated samples The L_∞ norm on the other hand is very sensitive to a large deviation of a single sample, since among all samples considered, it selects the one that yields maximal deviation The choice of these different norms is basically a matter of robustness to fluctuations and computational complexity.

Given a certain choice of norm (say the L_n norm), the hypothesis prediction stage would be identical to what has been previously described, and given a hypothesis H_0, H_0 would be accepted if and only if

$$d_{n,X} < \lambda \quad and \quad d_{n,Y} < \lambda$$

where λ is the chosen threshold on the sample deviations

3.2.5 Singular Cases

As was mentioned previously, the edge components are normalized in order to yield information which is contrast-independent and which is only direction-dependent In order to do so the edge components e_{x_i} and e_{y_i} at points $P(x_i, y_i)$ on the predicted line are divided by the edge magnitude N_i In some cases however, it could happen that there exist points P_i on a predicted line such that their edge magnitude be $N_i = 0$ This would correspond to the case where the predicted line spans a portion of a uniform region (i e a region with no contrast at all) This is what we call a 'singular case," since in such cases, the usual hypothesis verification tests cannot be performed any more The approach we have taken is to reject lines which have at least one point at which the edge magnitude is null In other words, the lines that are subsequently kept are not allowed to cross over any uniform region This, of course, prevents the proposed line detector from having any interpolatory capability and a contour line broken into two pieces by a uniform intensity patch will not be detected On the other hand, a predicted line might very well be accepted, depending on the chosen tolerances, even if it covers regions having points of different gradient orientations as long as the gradient magnitude does not vanish at any of the points. It will of course be argued whether the proposed scheme is satisfactory We believe it is, owing to its simplicity, since lines which are artificially broken up due to noise could be merged together at a higher processing level

3.2.6 Complexity Analysis

We shall now study the computational complexity involved in performing line detection with the proposed algorithm. The complexity involved in the preprocessing stages (edge enhancement and edge detection) is not included in this analysis We thus assume that we are given K edge elements Computing the normalized gradient involves 1 addition operation, 2 multiplications, 2 divisions and 1 square-root operation for every edge element considered Assuming we are looking for lines formed of N pixels and assuming we are using the d_{2X} and d_{2Y} deviation criteria, $2(N-1)$ addition operations and 2 divisions are involved in calculating the sample mean, and $4N-2$ addition operations, $2N$ multiplications and 2 division operations are necessary for calculating the sample variance, for every edge element considered In total therefore, given K edge elements and lines of length N, we have $3K(2N \quad 1)$ addition operations, $2K(N+1)$ multiplications, $6K$ divisions, and K square-root operations involved The computational complexity of the algorithm is thus seen to grow linearly in the number of edge elements and in the length of the lines to be detected

3.2.7 Comparison with Burns et al.'s Algorithm

Now that the main body of the algorithm has been presented, it would be interesting to perform a comparison with Burns et al.'s algorithm which was introduced as being the closest to the algorithm presented in this thesis While both our algorithm and theirs are based on gradient orientation information, the main discrepancy comes from the fact that they do not make full use of all the constraints involved, the main one being the following: if a contour line segment is supposed to go through an edge point, then the gradient orientation at that edge element should be orthogonal to the contour line. It is this very same observation that allows us to make a prediction on the precise location and orientation of the line segment to be detected. The second constraint comes from the fact that in order for the predicted line segment to actually overlap with an image contour line,

the first constraint must be met (in a loose sense) at all the points on this predicted line This is the information that we use in the verification process Burns et al 's algorithm makes partial use of the second constraint by accepting in a line support region only those edge elements that have similar orientations The information it does not use, however, is that each edge element constrains the underlying contour line to have a unique position and orientation Thus, they end up doing a type of region growing operation without ever considering the geometrical structure of the region being formed

3.2.8 Sequential Pruning

Note that the hypothesis verification procedure outlined above will validate the existence of a line at a given point based only on the statistical tests mentioned, and therefore overlapping between line segments are to be expected In order to reduce this overlap, and hence yield a smaller and more tractable set of line segments for subsequent analysis, an additional test could be incorporated in the verification procedure so as to discard line segments which have too much overlap with previously detected ones This test could be conducted as follows for every hypothesized line segment S which also satisfies the mean and variance tests, let $\{(x_i, y_i)\}$ be the set of points which constitute S Furthermore, let N_S be the number of points in S which are part of previously detected line segments *of the same order* Assuming the order of the line segments to be n, the number of points on each line is then $N = 2n+1$ Thus the ratio N_S/N is the percentage of points in S which are part of previously detected line segments A hypothesized line segment S which also satisfies the mean and variance test could thus be validated if the above ratio is less than some specified threshold (i.e. if the overlap between the line segment considered and previously detected ones is less than some amount) and rejected otherwise While the linear structures present in the image are captured in the initial hypothesis test, the net effect of adding this extra test is that a smaller number of line segments are typically detected. This simplifies subsequent interpretation tasks.

3.2.9 Resolution Trees and Multiple-length Line Detection

Now that the line detection algorithm has been presented, we investigate a possible representation for the set of line segments that are present in a given picture. This representation can be based either on the orientation of the line segments, or on their length. Representing the line segments in the image in terms of their orientation can be of interest whenever attention is to be focused on those line segments that have specific orientations. Representing line segments based on orientation reduces to assigning them to particular orientation classes. A property of this representation is its lack of ambiguity, since no two line segments that share a common set of pixels can have distinct (to within a certain quantization factor) orientations.

Representing line segments based on their length is another representational scheme, which is more suited for cases where the line segments are to be matched to a specific model. In this case, information concerning the length of the different line segments is used in order to verify the validity of candidate matches. Representing line segments based on their length is again a grouping problem, but unlike the previous one, it is inherently ambiguous, since a certain line segment can very well be composed of a number of smaller segments. The problem then is to choose a representation that overcomes the ambiguity associated with redundant segments.

The representation we chose is based on a tree structure. The root R of the tree is the original image. The first level of nodes, starting from the root, is formed of line segments S_i^m constituted of m pixels. Each node is itself expanded into a subtree, where each subnode corresponds to a line segment S_j^n of inferior length, with the additional property that the segment associated with the subnode is included in the segment associated with the node. In other words, S_j^n is a subnode of the node S_i^m if and only if:

$$n < m, \quad , S_j^n \subset S_i^m$$

In this context, the inclusion of S_j^n into S_i^m is equivalent to the number of pixels common both to S_j^n and a neighborhood $\mathcal{N}(S_i^m)$ of S_i^m being superior to a certain fraction λ of the

total number of pixels that constitute S_j^n In other words,

$$(S_j^n \subset S_i^m) \Longleftrightarrow (Card(S_j^n \cap \mathcal{N}(S_i^m)) \geq \lambda Card(S_j^n))$$

where $Card(S)$, the cardinality of a set S, is defined as the number of sample points in S In subsequent experiments, the value for λ was chosen as being a half It is to be noted that a higher value for this coefficient would not yield the desired results due to the fact that in practice, similar line segments do not always coincide In addition, the reason for evaluating inclusion over a neighborhood of the larger segment rather than the segment itself stems from the fact that the edges are not of the ideal step edge type and have a certain spatial spread The neighborhood $\mathcal{N}$ of each line segment is defined as the region formed by thickening the line segment width by one pixel on each side This is equivalent to convolving the image of the line with a spatial averaging mask and then retaining only the pixels that have an intensity value above some predetermined threshold

The process of expanding nodes into subnodes is done recursively until a minimal length of line segment is achieved The segments having minimal length are the leaves of the tree (see Figure 3 6)

This representation is analogous to the one used in representing images at multiple resolutions, which, owing to its geometrical arrangement is often referred to as a "resolution pyramid" In a similar way, we refer to the problem of detecting lines of different length as that of detecting lines at different resolutions in the resolution tree A major advantage of this representation lies in its ability to distinguish line segments based on their significance Intuitively, a line segment which is included in another line segment of superior length does not have the same significance as another line segment which is *not* included in any other line, since this would amount to redundant information In order to remove these redundancies and ambiguities, line segments having low significance are discarded The process of removing lines of low significance is called "pruning"

Pruning the resolution tree is done by removing those subnodes which are not *directly* linked to the root. In other words, line segments which are part of longer segments

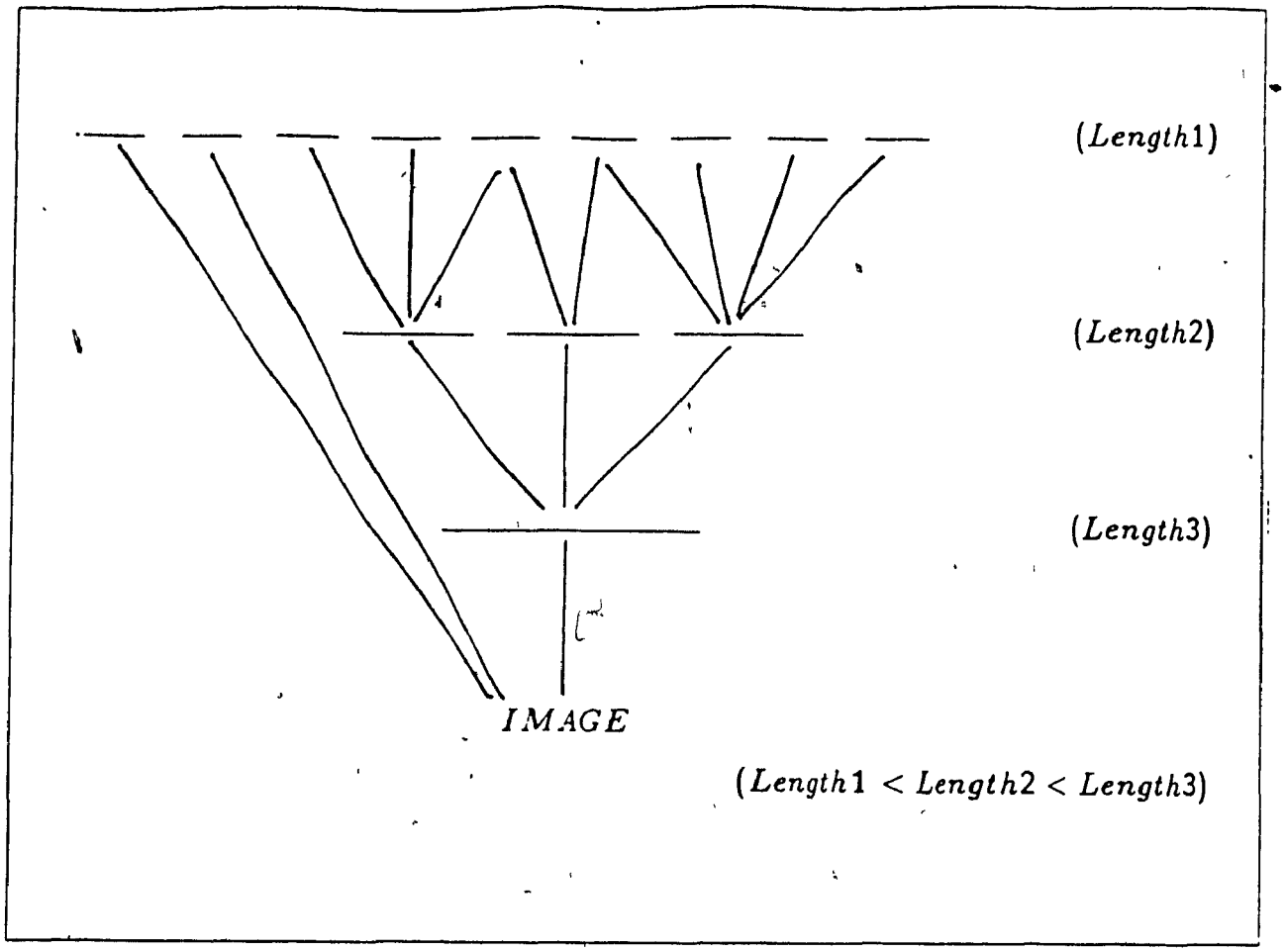


Figure 3.6 The resolution tree

(i.e., subnodes of the resolution tree which are directly connected to other subnodes rather than to the root) are discarded. The resulting structure is a tree having one root, and leaves at different levels, indicating the different lengths of the line segments associated with the leaves. This remaining structure is in a sense a best piecewise fit of line segments to the image, since all the partial fits have been discarded in the pruning process.

To perform pruning, we do as follows: first define the different lengths at which the line segments are to be detected. Then find the lines, starting with the longest, and proceeding to the shortest. With each line detected, flag the corresponding pixels that constitute it. Thus, when evaluating whether a line segment is to be retained, it is not only necessary to perform the statistical tests as described in section 5, but we also must take into account whether the line segment considered is included in previously detected

line segments of longer length Since the computational cost of evaluating inclusions of line segments in other line segments is considerably less than the cost of performing statistical tests on the lines, pruning could be used to reduce the number of predicted hypotheses that have to be verified

It should be noted that the resolution with which line segments of different length can be detected depends solely on the number of different levels in the resolution tree, and that the larger the chosen number of levels in the resolution tree, the finer the representation of line segments in the original picture

Chapter 4 Experimental Results

The line detection algorithm has been successfully applied to a wide variety of digital pictures. The robustness of the algorithm is inherent in its two-step procedure: while the prediction of hypotheses is essentially local to a picture element the verification of the predicted hypotheses involves statistical tests over a selected neighborhood In what follows, a number of experiments are illustrated In the ones involving statistical tests, the 95% confidence intervals on the population statistics are computed (i.e $\alpha = 0.05$) The purpose of these experiments is to show how the performance of the proposed algorithm depends on the choice of its associated parameters The first test image, shown in figure 4.1, is that of a capacitor

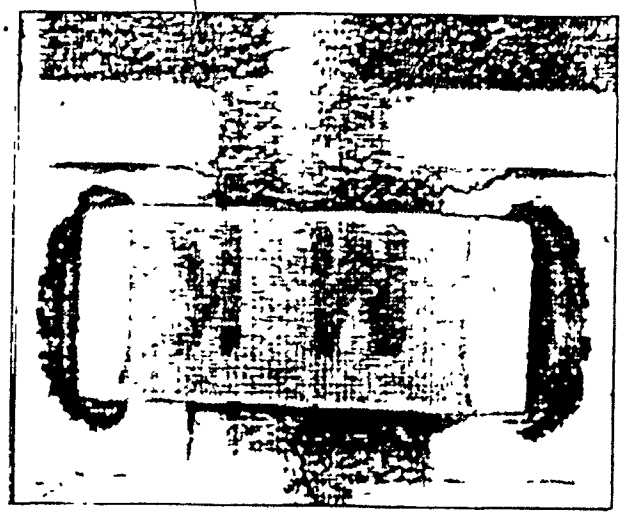


Figure 4.1 Original image

mounted on a ceramic substrate. This image is chosen not only because it is rich in contour lines but also because the wide diversity of linear features present in it would illustrate well the performance of the proposed algorithm in real images

The experiments on the line detection algorithm are performed as follows. first, the effect of performing sequential pruning is highlighted on the final result Different parameters for this pruning are chosen, and the one which yields satisfactory results is chosen for further experiments Then, using the statistical model, the result of assigning different values to the threshold variance σ_T^2 and to the threshold mean difference Δ_μ is presented Subsequently, different line lengths are chosen for the line detection algorithm based on the statistical model, and the results are discussed The next group of experiments is performed on the line detection algorithm without use of the statistical model Rather, the simplified criteria presented in section 3 2 4 are used, and the associated results are shown Following this, the use of the resolution tree is presented and a best fit of lines (within the lengths considered) to the image is performed Finally, the performance of the algorithm is illustrated on some images of outdoor scenes

Before, however, the preprocessing steps associated with the algorithm are performed

4.1 Image Preprocessing

Figure 4.1 shows the picture of a capacitor mounted on a ceramic substrate In this picture, as well as in all subsequent ones, the gray levels range from 0 to 63 The linear features in this image are those of the delimitating boundaries of the capacitor, and also those of the conductive traces on the substrate The nonlinear features present in the image are mainly due to surface texture. Figure 4 2 shows the same picture after edge enhancement using the Sobel masks and thresholding with a threshold of 10, followed by thinning;

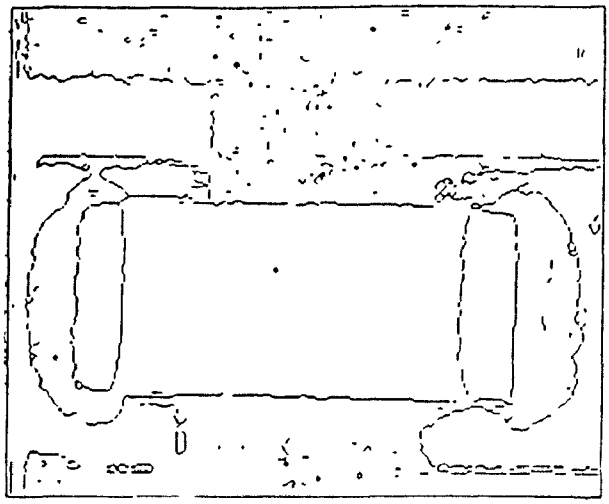


Figure 4.2 Edge elements

In other words, in a first step, all edge elements having a gradient magnitude greater than or equal to 10 have been assigned the gray value 63, while others have been assigned the gray value 0 In a second step, the resulting binary image is thinned in order to reduce the spatial spread of diffused edges The thinning algorithm used is Hilditch's sequential algorithm for thinning binary images[23] Thus, only those edge elements that have survived both the thresholding and the thinning steps are considered by the line detection algorithm

As observed in the picture, only a small portion of the edge elements correspond to linear features, and this can be attributed to the large amount of surface texture Figure 4.3 shows the orientations of the edge elements that have been retained after intensity thresholding. These orientations are symbolically displayed by short line segments of equal length having the direction of the gradient at the point from which they emerge

4.2 Sequential Pruning

In this experiment, line detection is performed using the statistical model. The length of the line is chosen to be 55 pixels, and the threshold values are chosen to be $\sigma_T^2 = 0.2$ and $\Delta_\mu = 0.2$. Three different experiments are performed in order to highlight

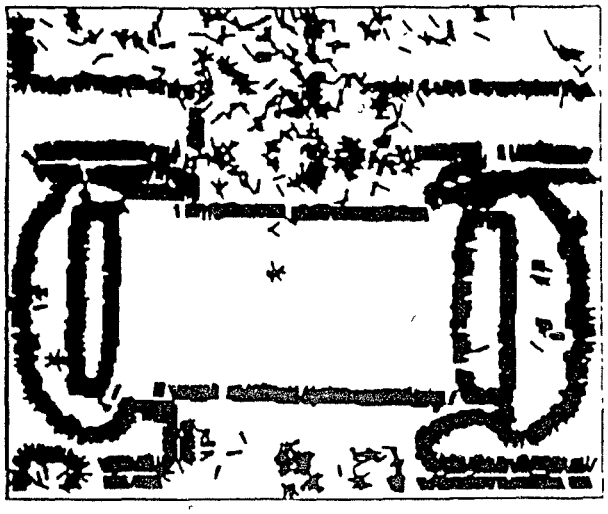


Figure 4.3 Gradient vectors

the use of sequential pruning In the first experiment lines are detected without any sequential pruning being performed The result is shown in figure 4 4

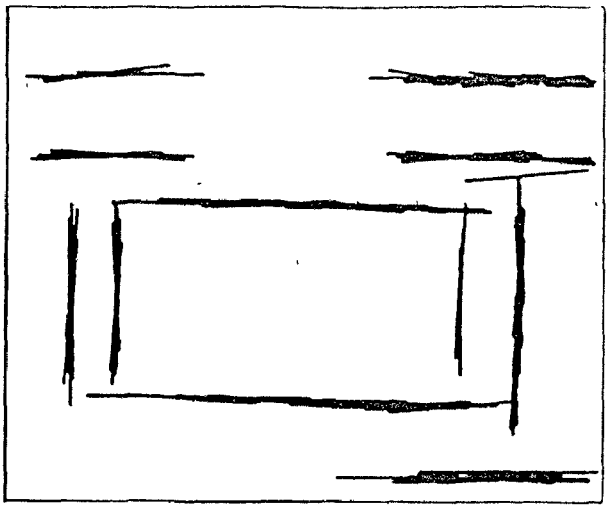


Figure 4.4 Line detection with no sequential pruning

In this case, 225 line segments were found Clearly, for the type of picture considered and for the length of line chosen, this number is overwhelming In figure 4 5, lines are detected *with* sequential pruning and such that each line segment detected shares at most 10 percent of its pixels with previously detected line segments

In this case, only 34 line segments were found. Note that although the number of line segments detected is in this case much inferior to the 225 that were previously found,

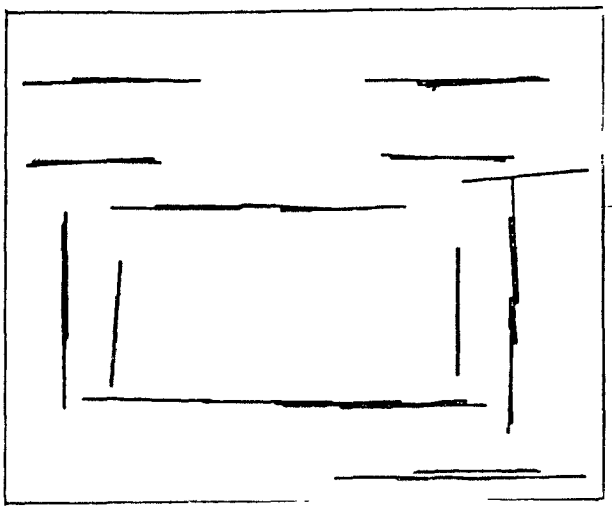


Figure 4.5 Line detection with sequential pruning (10 % overlap)

the linear structures in the image are accurately captured. In figure 4 6, finally, the result of sequential pruning is displayed, where no two line segments are allowed to intersect

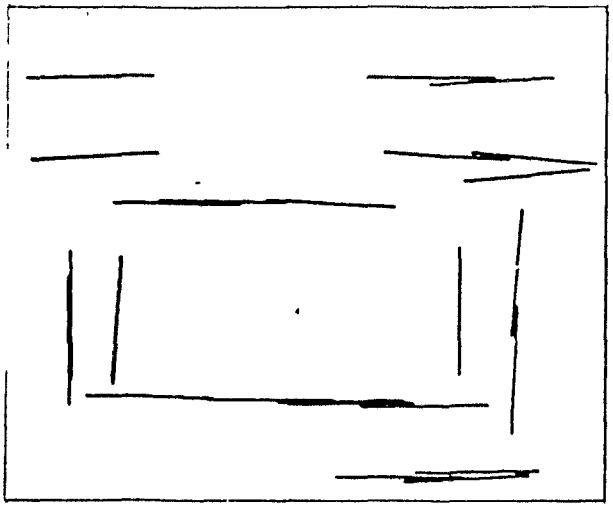


Figure 4.6 Line detection with sequential pruning (no overlap)

This further reduces the number of detected lines to 24 In this case, a number of linear features are missed.

Since the purpose of this line detection algorithm is to act as a preprocessing step for higher level interpretation stages, it is of importance to reduce the amount of data that is to be fed to higher levels. Sequential pruning then becomes necessary insofar as, with the proper choice of parameter, the number of line segments detected is considerably

reduced and the most of the linear features of the image are captured In all subsequent experiments, therefore, sequential pruning *will* be performed, and a line segment will be detected only if it shares less than 10 percent of its pixels with previously detected lines

4.3 Statistical Model: Choice of Parameters

In the following experiments, the influence of the thresholds σ_T^2 and Δ_μ on the detection of lines is shown Remember that σ_T^2 sets a limit on the uniformity of the orientations of the edge elements that are on the predicted line segment, while Δ_μ sets a limit on the deviation of the orientation of these edge elements from the orientation of the line segment that they are supposed to constitute Three types of errors might occur, depending on the choice of σ_T^2 and Δ_μ If both are chosen conservatively then the variance in orientation of the edge elements forming the detected line segments is constrained to be small and the average orientation of these edge elements is constrained to be close to that of the line segment itself This is precisely the case where, owing to the small variance, the normality assumption holds If, however, σ_T^2 is given a large value and Δ_μ is set conservatively, then the predicted line segments will be accepted based only on the *average orientation* of their edge elements, regardless of their uniformity Conversely, if σ_T^2 is set conservatively and if Δ_μ is given a large value, then the line segments are accepted based only on the uniformity of the orientation of the underlying edge elements and regardless of their actual orientation Finally, if both σ_T^2 and Δ_μ are assigned large values, predicted lines are accepted even if they do not reflect the presence of underlying linear structures

Figure 4.7 shows the result of detecting lines 55 pixels long with $\sigma_T^2 = 0.1$ and $\Delta_\mu = 0.1$.

Also bear in mind, that the normalized edge components span the range $(-1, 1)$ The estimated deviation allowed with this choice of parameters is then close to 15 percent while the deviation from the mean is constrained to lie within 5 percent As a result, 27 line

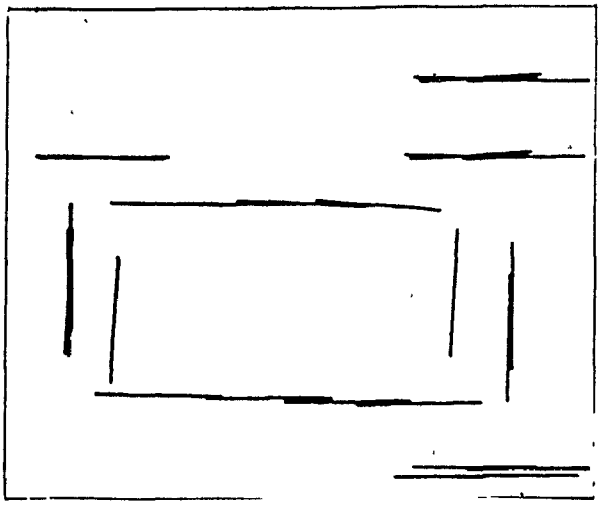


Figure 4.7 Line detection with $\sigma_T^2 = 0.1$ and $\Delta_\mu = 0.1$

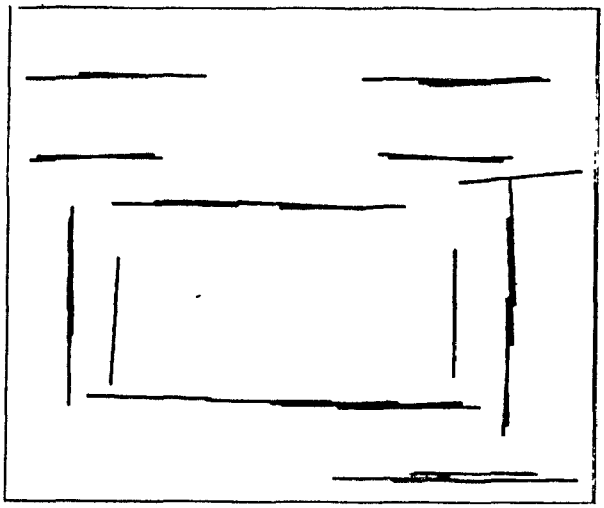


Figure 4.8 Line detection with $\sigma_T^2 = 0.2$ and $\Delta_\mu = 0.2$

segments are detected. Figure 4.8 shows the result of detecting lines of the same length and with $\sigma_T^2 = 0.2$ and $\Delta_\mu = 0.2$

In this case, 34 line segments are detected. The additional lines that are detected are "more noisy" than the ones found with the previous parameter settings and hence could not satisfy the previously required conditions. Figure 4.9 shows the result of detecting lines with $\sigma_T^2 = 0.3$ and $\Delta_\mu = 0.3$

In this case 42 line segments are detected, some of which do not correspond to linear features. In effect, the thresholds have been relaxed to the extent that line segments

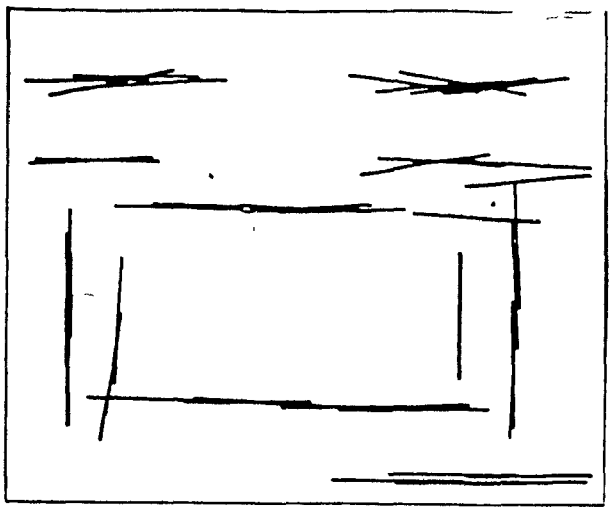


Figure 4.9 Line detection with $\sigma_T^2 = 0.3$ and $\Delta_\mu = 0.3$

are accepted even if the underlying pattern they cover is not perceived as being linear This artefact is even more pronounced in figure 4 10. where values of $\sigma_T^2 = 0.5$ and $\Delta_\mu = 0.5$ are chosen

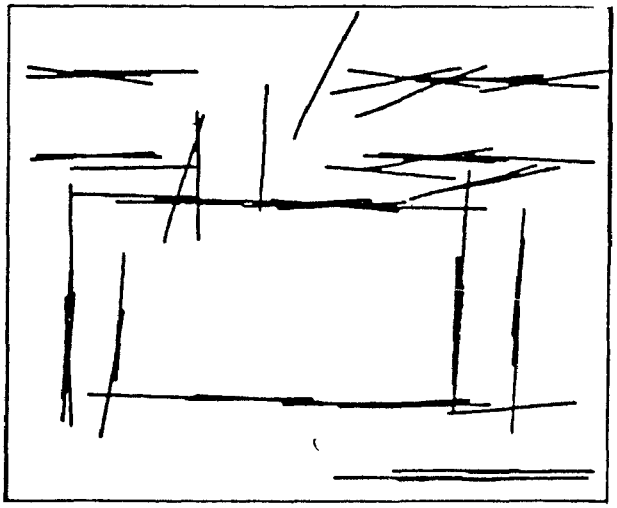


Figure 4.10 Line detection with $\sigma_T^2 = 0.5$ and $\Delta_\mu = 0.5$

In total. 55 line segments are detected. and among these. many which are nothing more than a set of randomly oriented edge elements

Based on the previous experiments. values of $\sigma_T^2 = 0.2$ and $\Delta_\mu = 0.2$ were found to yield reliable line detection. One should not forget that these values depend to a large extent on the picture being processed Had the image been noise-free. much smaller

values of these two parameters could have been safely chosen. Since the image is not ideal, however, tolerance to deviations has to be embedded in the values assigned to these thresholds.

4.4 Statistical Model: Choice of Line Length

In this experiment, we will investigate the dependence of the results on the length of line that is specified Of course, the choice of the proper line lengths is closely related to the a priori knowledge one has about the image to be processed Different line lengths, however, will induce different results and these will be examined now

"Based on the previous results, we set the threshold variance to $\sigma_T^2 = 0.2$ and the threshold mean difference to $\Delta_\mu = 0.2$ We first detect lines of length 155 The result is shown in figure 4 11, and only 2 lines are detected which correspond to the long sides of the capacitor.

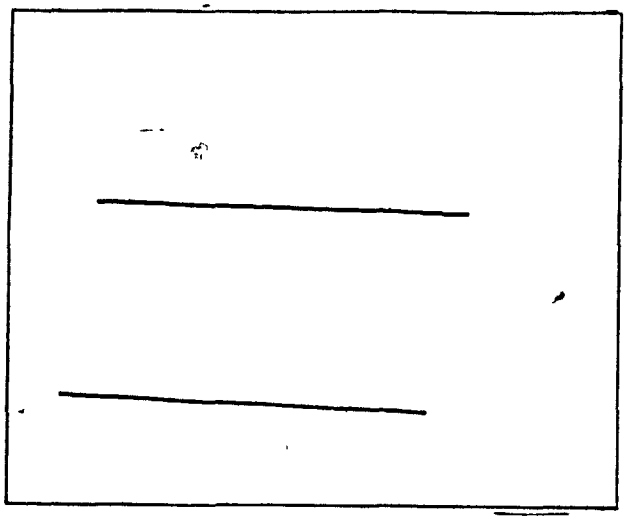


Figure 4.11 Line detection with length 155.

In figure 4.12, the result of detecting lines of length 75 pixels is shown.

In this case, the two sides of the capacitor are again found, in addition to other contour lines which were not accomodated by the previous line length This simple example

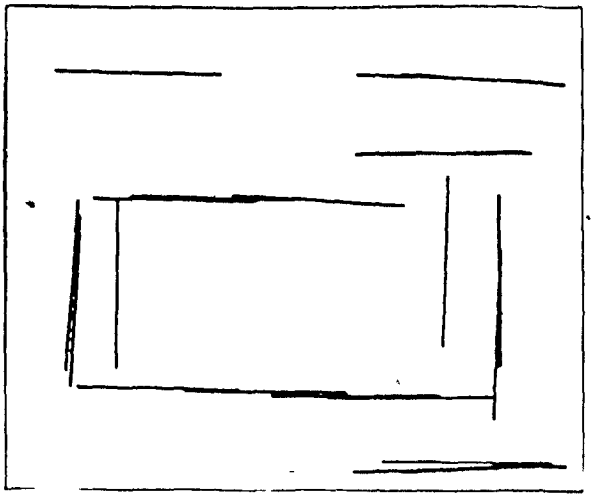


Figure 4.12 Line detection with length 75

illustrates well enough the concept of significance of detected lines that was introduced in the previous chapter Conceptually, the lines of length 75 that overlap with the previously detected lines of length 155 do not have the same significance as the ones that do not overlap, since in the former case the lines of length 155 yield a more complete fit than the lines of length 75 This point will be illustrated later on with an example implementation of the resolution tree Figure 4 13 shows the result of line detection with a length of 35

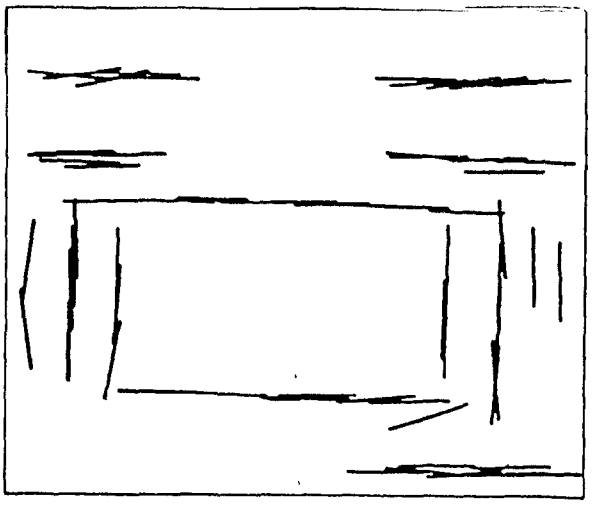


Figure 4.13 Line detection with length 35

More and more fine detail is captured as compared to previous line lengths. Figure 4.14, finally, shows the result of line detection with a specified length of 15 pixels.

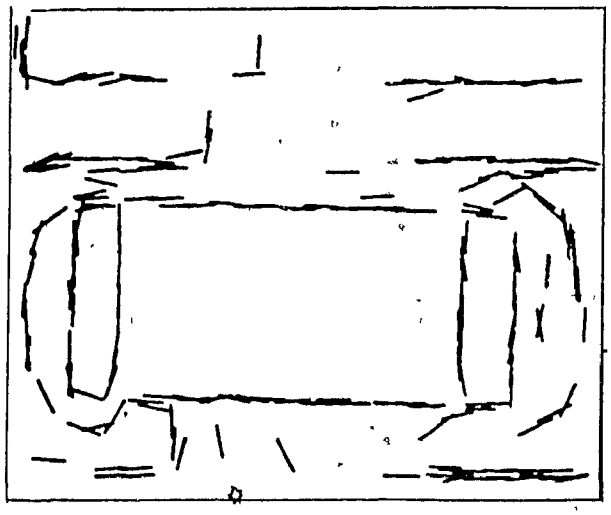


Figure 4.14 Line detection with length 15

At this point we could ask ourselves the following question: is it better to detect long lines first and then specify successively smaller and smaller lengths (in order to catch more and more fine detail), or is it better to detect very short segments once and for all, and then group them together wherever possible in order to form longer segments ? The first approach suffers from the computational complexity involved, since a variety of different lengths would have to be specified The second approach however, suffers from the noise sensitivity inherent in the detection of short line segments A short line segment is, by definition, formed of a small number of samples (or pixels) Using the statistical model therefore, small deviations in any of the samples might considerably affect the estimated intervals on the underlying population statistics, and this extrapolation step might be very noise sensitive. Of course this problem is not as accute with the other suggested scheme which is based solely on sample measurements The second problem with detecting short segments lies in the obervation that if, on a long line segment, noisy samples are not randomly distributed but are rather spatially grouped, a short segment can not be detected at that position Despite all these problems however, an algorithm is suggested in the conclusion of this thesis for merging small line segments into longer ones while overcoming the noise sensitivity associated with the detection of short lines. Detecting long line segments does not present the same kind of noise sensitivity problems, since averaging is performed over a larger number of observed samples The main problem

associated with detecting long lines, however, is inherent in the fact that the orientation of the predicted line is based on the gradient orientation of a single edge element, and slight deviations in the gradient orientation could yield large deviations of the predicted line from the true contour line and as a result, the line will not be detected

4.5 Non-Statistical Model: Choice of Deviation Criterion

The previous hypothesis verification criterion was based on the premise that the normalized edge elements were normally distributed As was pointed out in section 3 2 4, this assumption is valid only for small variances, and cannot be deemed true for arbitrary threshold values A new set of hypothesis verification criteria was therefore proposed and which were not based on interval estimation

In this group of experiments, we will investigate the choice of an appropriate deviation criterion, from the ones proposed in section 3 2 4 of this thesis The length of the lines to be detected will be set to 55 pixels The deviation criteria considered are d_1, d_2 and d_∞ which are based on the L_1, L_2 and L_∞ norms respectively As was pointed out previously, the L_1 norm is equivalent to an averaging operation, while the L_∞ norm is equivalent to maximum value selection The deviation criterion d_1 is thus an *average deviation* measure, while the criterion d_∞ is a *maximum deviation* measure d_1 will thus exhibit dependence on group deviations, while d_∞ will exhibit dependence on sample deviations Any choice of norm between these two extremes will thus be a tradeoff between noise immunity and sensitivity to changes in orientation In what follows, the threshold on the deviation in the $x-$ and $y-$ edge components will be denoted by λ

The first experiment is performed using the d_{1X} and d_{1Y} criteria (as defined in equation 3 35) with $\lambda = 0.1$. In other words, the tolerated average deviation shall not exceed 5 percent of the full range. As a result, only 9 lines are detected, as shown in figure 4.15.

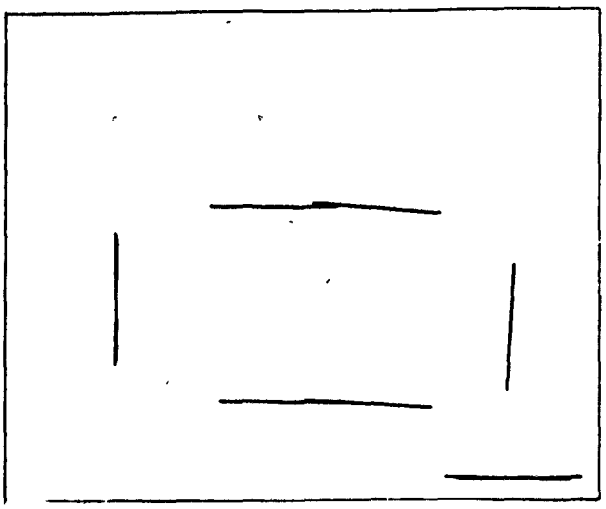


Figure 4.15 Line detection with $\lambda = 0.1$

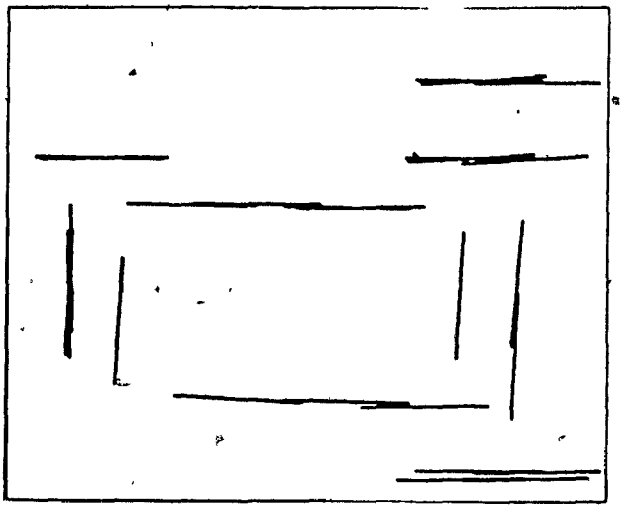


Figure 4.16 Line detection with $\lambda = 0.2$

Relaxing this condition and doubling the allowed deviation yields 26 lines (figure 4.16)

While the lines previously found were seen as being "perceptually straight," increasing the tolerated deviation on the normalized edge components results in additional lines which could be qualified as "perceptually noisy." Increasing the tolerance to $\lambda = 0.5$ finally, yields additional lines which do not coincide with underlying contour lines (figure 4.17).

We now use the d_{2X} and d_{2Y} criteria (see equation 3.34) with $\lambda = 0.1$. Since

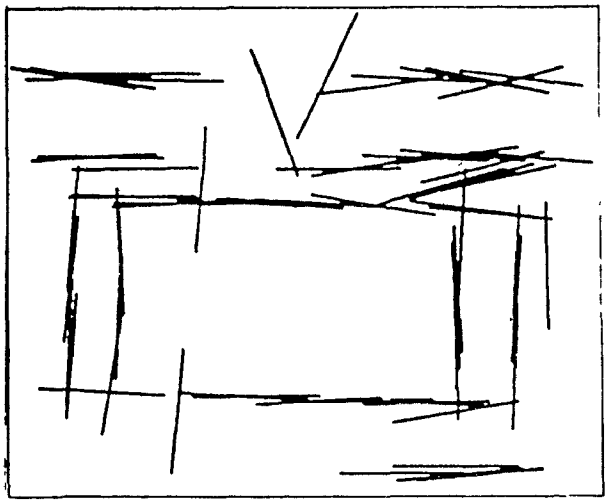


Figure 4.17 Line detection with $\lambda = 0\,5$

less averaging is performed using these criteria (as opposed to d_{1X} and d_{1Y}) and since, as a result, more sensitivity is exhibited to individual sample deviations it would be reasonable to expect fewer lines to be found and indeed, only 2 are detected (figure 4 18)

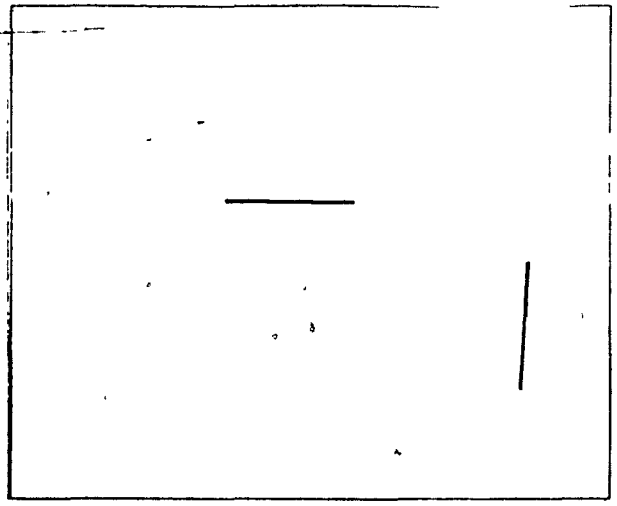


Figure 4.18 Line detection with $\lambda = 0\,1$

Doubling the allowed deviation to $\lambda = 0.2$ yields 18 lines which are seen to be less noisy than the ones found with the d_{1X} and d_{1Y} criteria (figure 4 19)

Finally letting $\lambda = 0.5$ yields 37 lines, among which many exhibit noticeable deviations from the underlying contour lines which they are supposed to match (figure 4.20).

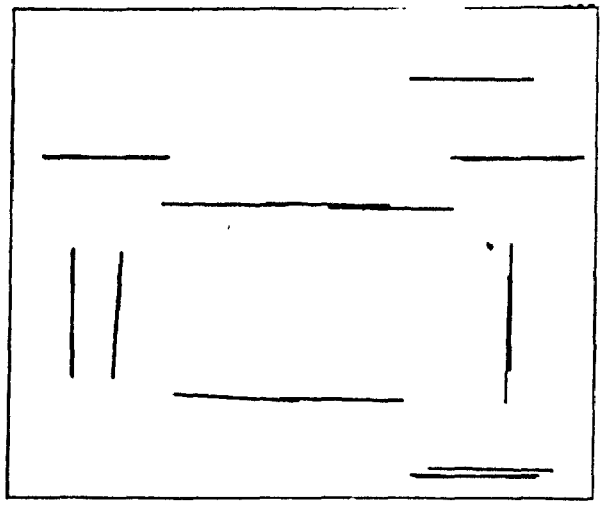


Figure 4.19 Line detection with $\lambda = 0\,2$

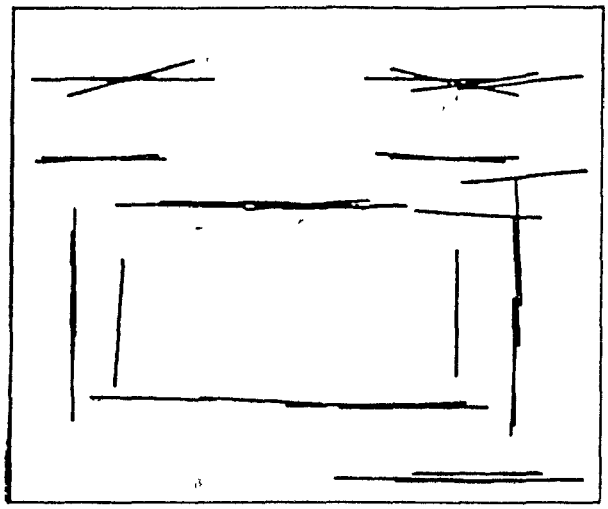


Figure 4.20 Line detection with $\lambda = 0\,5$

The last criterion considered is based on the L_∞ norm and is formed of $d_{\infty X}$ and $d_{\infty Y}$. This criterion is the most sensitive to individual sample deviations since no averaging operation whatsoever is performed. This criterion is thus the most noise sensitive, and indeed, no lines are found with $\lambda = 0.1$ and $\lambda = 0.2$. This shows that in the lines previously found with other criteria, there were samples exhibiting individual deviations larger than what was tolerated, but these deviations were averaged out in the set of samples considered. Setting $\lambda = 0.5$ however, yields 16 lines all of which are seen to match the least noisy underlying contour lines (figure 4.21).

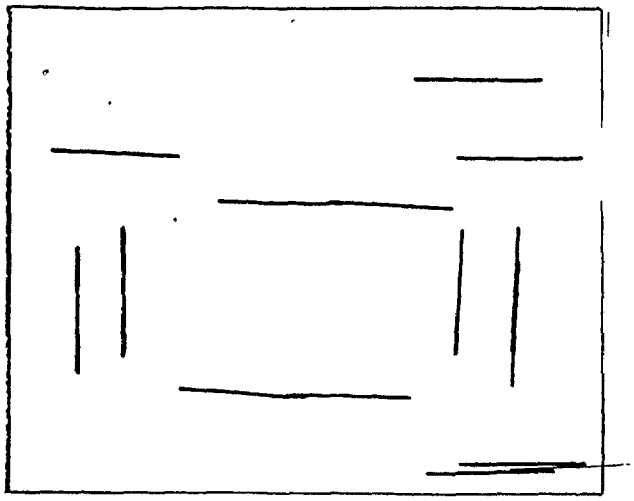


Figure 4.21 Line detection with $\lambda = 0.5$

Based on the previous experiments, we will adopt the d_{2X} and d_{2Y} criteria for subsequent experimentation This choice is dictated by their predicted and observed tradeoff between immunity to noise and sensitivity to changes of orientation

4.6 Pruning the Resolution Tree

In this section, we will perform experiments on the resolution tree, which was introduced in section 3 2 9 as a useful data structure for highlighting redundancies between detected lines of different length The different lengths that the lines will be detected at are 155, 75 and 35 In all cases, the maximum tolerated deviation λ is set to $\lambda = 0.4$. Figure 4.22 shows the detection of lines of length 155

As expected, only the two main sides of the capacitor have been detected. In figure 4.23, the result of detecting lines of length 75 *after pruning* is shown

As expected, overlap between lines of length 75 and the ones previously found (i.e. lines of length 155) have been reduced to a minimum Similarly, figure 4 24 shows the result of line detection with length 35 and after pruning is performed

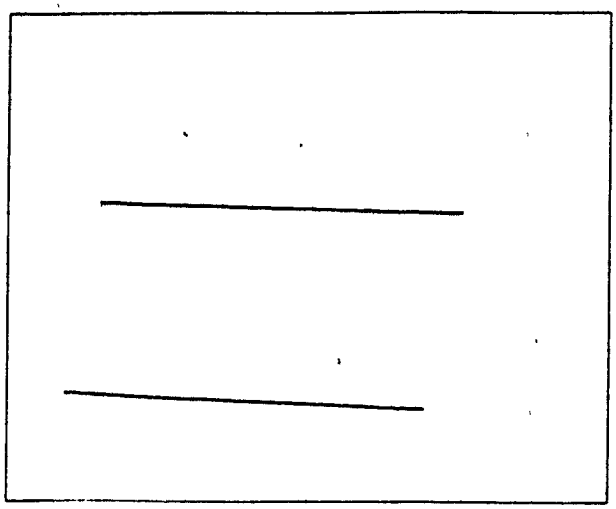


Figure 4.22 Line detection with length 155

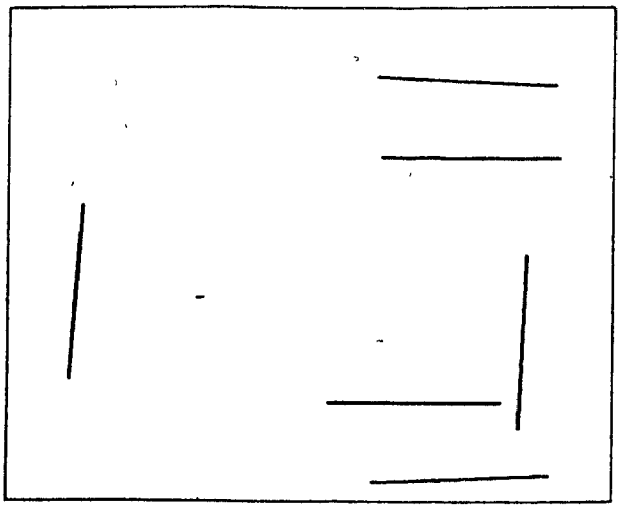


Figure 4.23 Line detection with length 75

Again, redundancies between these lines and the ones found previously (i.e. lines of length 75 and 155) have been reduced to a minimum In figure 4 25, the last three images are overlapped for the sake of clarity

As observed, redundancies between lines of different length have been reduced, and each line segment in the resolution tree is a best piecewise fit to the image, for the different line lengths that have been selected.

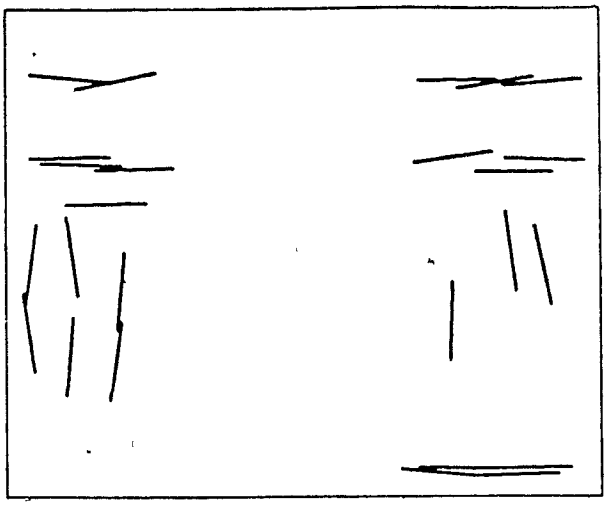


Figure 4.24 Line detection with length 35

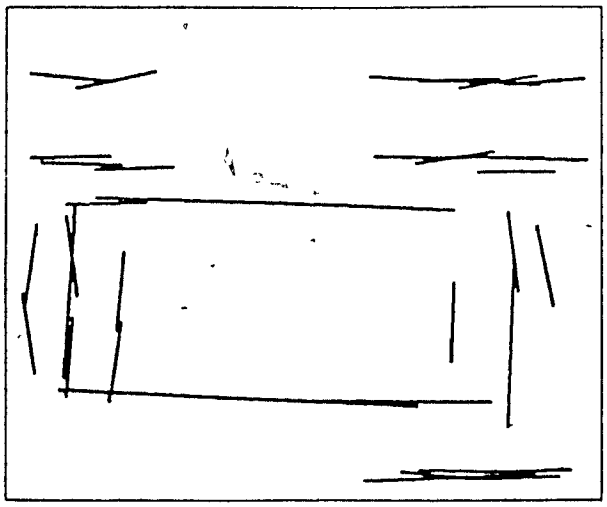


Figure 4.25 Pruned resolution tree

4.7 Example Outdoor Scenes

Figure 4 26 shows an outdoor scene

The linear features that are present in this picture correspond to the car chassis, and part of the projected shadows In Figure 4 27, the result of line detection is displayed with $\lambda = 0.4$ and a line length of 55.

As expected, the main linear features are captured. Another outdoor scene image is shown in figure 4.28.

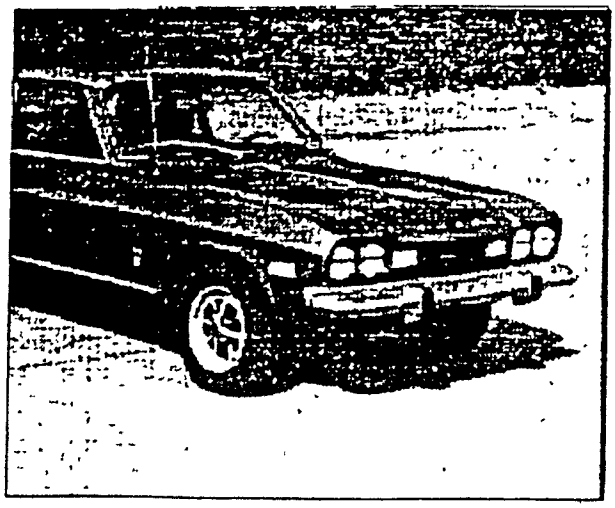


Figure 4.26 Example outdoor scene

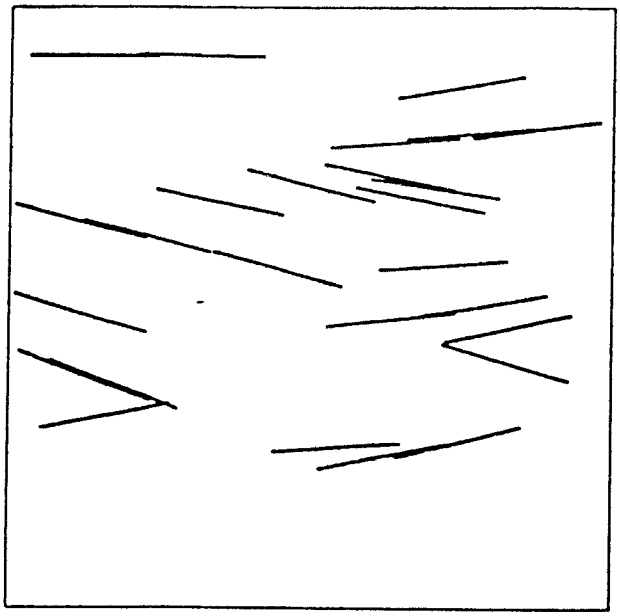


Figure 4.27 Line detection with length 55

Line detection is performed on this image with again the same parameters, and the result is shown in figure 4.29.

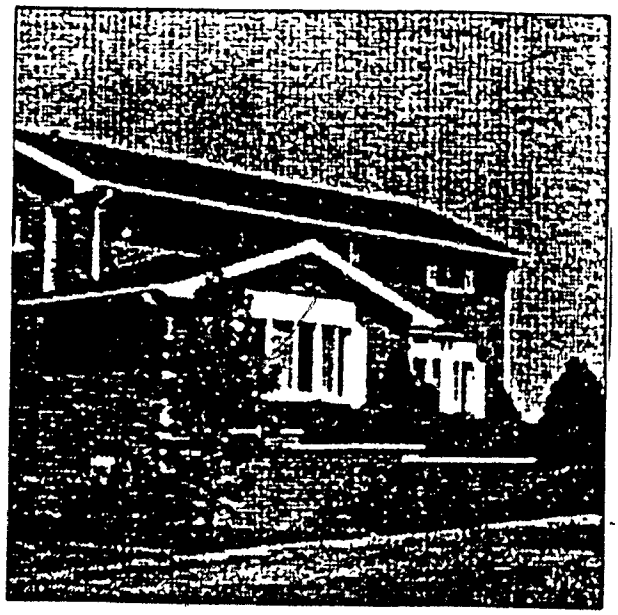


Figure 4.28 Example outdoor scene

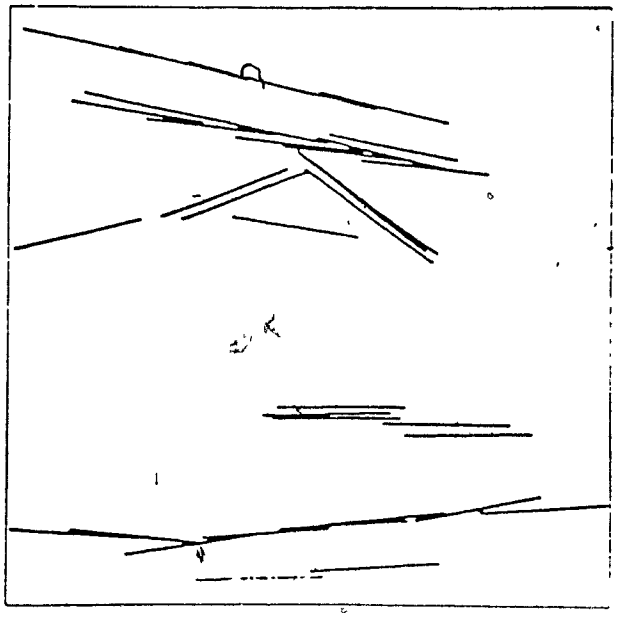


Figure 4.29 Line detection with length 55

Chapter 5 Conclusions

In this thesis, an algorithm was presented that detected line segments in intensity images. This algorithm was shown to be based on the hypothesis prediction/verification paradigm, and was found to be both computationally efficient and robust to fluctuations in the input data. The algorithm works as follows for every edge element predict a line segment of specific length centered at that same edge element, and verify this prediction using statistical tests on the samples that constitute this predicted line The prediction is done using the gradient orientation at the edge element, and hypothesis verification is performed using the gradient orientation of all edge elements that are on the predicted line. In addition to the basic algorithm, a data structure for representing the detected lines was introduced in order to reduce the redundancies between lines of different length Experiments were performed on the algorithm using both the statistical and the non-statistical models, highlighting the dependence of the algorithm on its associated parameters Additional experiments were also performed with the resolution tree, and the result of pruning it was shown to yield a best linear fit to the original image from within the line segments that had been originally considered. Finally, the algorithm was successfully tested on images of outdoor scenes.

Two possible extensions could be envisaged for this algorithm: the first relates to the grouping of lines that have been detected, and the second relates to the detection of curves (of a restricted type) using the same paradigm that has been adopted here for the case of line detection. In what follows, each of these ideas is briefly discussed.

5.1 Line Merging

The algorithm, as presented in this thesis, detects lines of a specified length. In many practical cases, no a priori knowledge of the possible line lengths in the picture is available The resolution tree was thus introduced as a means of representing lines based on their length in order to yield a best assignment of lines to the image The drawback with this approach lies in the computational complexity involved since a large number of different lengths have to be specified for line detection before any useful result can be attained A possible alternative to the resolution tree is to detect lines only for one possible length, and then to group together those lines which are actually part of the same contour line

When are two line segments part of the same contour line ? When any group of points taken from any of the two lines are collinear This condition being necessary, is far from being sufficient however, since one also has to take into account the proximity of the two lines We thus have constraints on orientation and constraints on proximity

The algorithm for line merging, embedding the constraints mentioned above, can be efficiently framed into the hypothesis prediction/verification paradigm, and goes as follows consider merging every pair of lines if their normalized gradient components differ by less than a certain threshold value (which could be the same as the one used for detecting the lines in the first place) This means in effect that the lines are close to being parallel. One now has to verify that they actually originate from the same contour line To do so, predict the line formed by merging the pair being considered (through joining of their end-points), and compute the normalized gradient components of the new line just formed (figure 5 1)

Now perform a hypothesis verification test on that line, exactly in the same manner as the ones shown in chapter 3 of this thesis If the test fails, then the merge is not valid. If the test succeeds and the hypothesis is accepted, then the new line is kept.

The merging process is then iterated until no new merges can be performed.

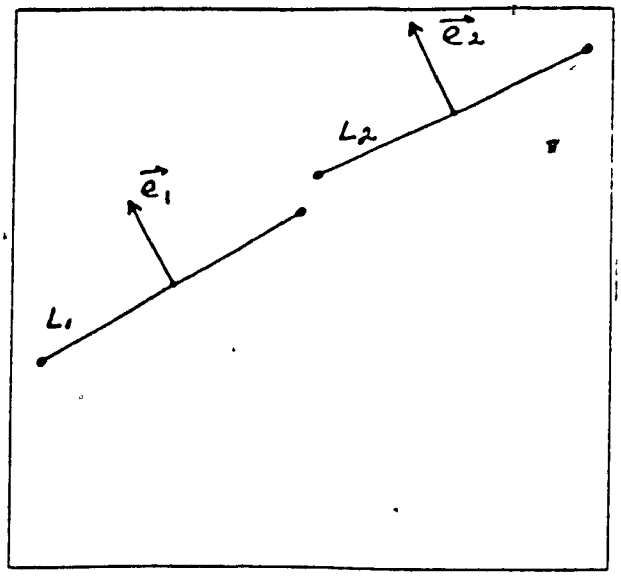


Figure 5.1 Line Merging

5.2 Curve Detection

The first question that ought to be answered at this point is· what is a curve? Simply stated, a (plane) curve is a mapping from the field of real numbers onto the two-dimensional plane This mapping could be arbitrary, and straight lines are just simple cases of plane curves Since curves exist in large varieties, it is necessary to somehow specify the kind of curve that we propose to detect We propose to detect only those curves which form arcs of circles Why only these and not others? Because we feel this to be the next logical step: the algorithm that was presented in this thesis detected straight lines, i.e. curves of zero curvature We now propose to detect arcs of circles, i e. curves of constant curvature. It can be seen, therefore, that line detection becomes a special case of the proposed algorithm for curve detection

The next question that ought to be answered is. how do we perform such a task? In order to do this, we shall draw analogs from what has been presented for line detection. Lines were detected based on two parameters· position and orientation. Arcs of circles will be detected based on three parameters. These are position, orientation, and curvature. Once these three parameters are computed for a certain edge element, it is

possible to predict a unique arc of circle having a prespecified length and which is centered at the edge element of interest Having the curvature of the predicted curve, one could easily compute its associated radius of curvature. From this radius and from the orientation and position of the edge element, one could compute the center of the predicted arc of circle Once this center is computed, the orientation and curvature values at the points on the predicted arc could be predicted, and these predicted values could be matched against the actual orientation and curvature measurements on those same points in order to verify the predicted hypothesis (figure 5 2) The main problem with this approach is, of course the reliable computation of a curvature estimate on which the curve prediction stage is based

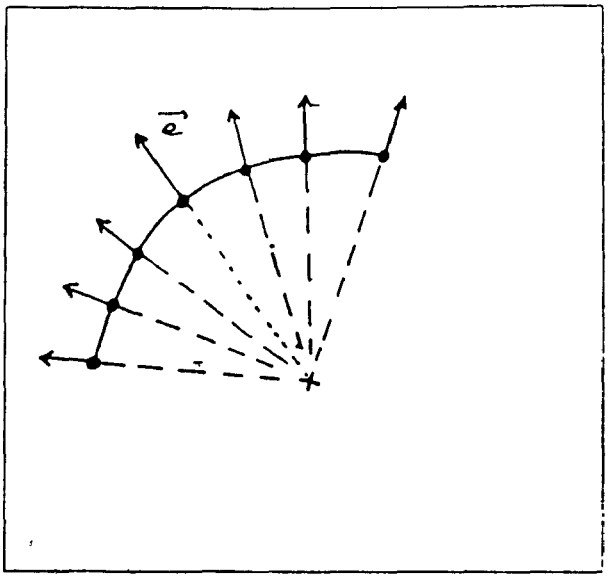


Figure 5.2 Curve Detection

The hypothesis prediction and verification stages are thus seen to be similar to what was presented for line detection except that additional steps are now included in order to accomodate curvature.

References

1. P. V. C. Hough. "Method and Means for Recognizing Complex Patterns." U.S. Patent 3069654. December 18, 1962.

2. R. O. Duda and P. E. Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures." *Communications of the ACM.* Vol 15. January 1972. pp 11-15.

3. S R Deans. *The Radon Transform and Some of its Applications.* Wiley-Interscience. 1983.

4. R. Hummel and S W. Zucker. "On the Foundations of Relaxation Labelling Processes." *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol 5. 1983. pp. 267-287

5. P. Parent and S W Zucker. "Trace Inference. Curvature Consistency and Curve Detection." McGill University Center for Research on Intelligent Machines Technical Report No CIM-86-3. June 1985

6. S. W. Zucker. R A. Hummel and A. Rosenfeld. "An Application of Relaxation Labeling to Line and Curve Enhancement." *IEEE Transactions on Computers.* Vol. 26. No. 4. April 1977. pp 394-403.

7. P. Parent and S W Zucker. "Radial Projection An Efficient Update Rule for Relaxation Labelling." Department of Electrical Engineering Technical Report No TR-85-15. McGill University. Montreal. June 1985

8. U. Montanari. "On the Optimal Detection of Curves in Noisy Pictures." *Communications of the ACM.* Vol 14. No. 5. May 1971. pp 335-345

9. Y. Shirai. "Analyzing Intensity Arrays Using Knowledge about Scenes." in *Psychology of Computer Vision.* P H. Winston. ed., 1975. pp. 93-113.

10. J B. Burns. A. R. Hanson and E. M. Riseman. "Extracting Straight Lines." *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol. 8. No. 4. July 1986. pp. 425-455

11. A.-R. Mansouri, A. S. Malowany and M D. Levine, "Line Detection in Digital Pictures: A Hypothesis Prediction/Verification Paradigm," Department of Electrical Engineering Technical Report TR-85-17, McGill University, Montreal, August 1985 Also *Computer Vision, Graphics and Image Processing* (to be published)

12. H G Barrow, "Interactive Aids for Cartography and Photo Interpretation" SRI Technical Report 5300, Menlo Park, California, November 1976

13. A. Rosenfeld and A C. Kak, *Digital Picture Processing*, Academic Press, New York, 1976

14. M. D. Levine, *Vision in Man and Machine*, McGraw-Hill, 1985

15. Y. G. Leclerc and S W Zucker, "The Local Structure of Image Discontinuities in One Dimension," Department of Electrical Engineering Report TR-83-19, McGill University, Montreal, 1983

16. L G Roberts, "Machine Perception of Three-Dimensional Solids," in *Optical and Electro-Optical Information Processing*, J T. Tippett, et al (eds), MIT Press, Cambridge, Mass., 1965, pp 159-197

17. R Nevatia, *Machine Perception*, Prentice-Hall, Englewood Cliffs, 1982

18. J M S. Prewitt, "Object Enhançement and Extraction," in *Picture Processing and Psychopictorics*, B S Lipkin and A. Rosenfeld (eds.), Academic Press, New York, 1970, pp. 75-149

19 J S Weszka, "A Survey of Threshold Selection Techniques," *Computer Graphics and Image Processing*, Vol 7, 1978, pp 259-265

20. O D Faugeras, N. Ayache, B Faverjon, "A Geometric Matcher for Recognizing and Positioning 3-D Rigid Objects," *Proceedings of the First Conference on Artificial Intelligence Applications*, 1984

21. P. H. Winston, *Artificial Intelligence*, Addison-Wesley, 2nd edition, 1984

22. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, New York, Academic Press, 1972

23. C J Hilditch, "Linear Skeleton from Square Cupboards," *Machine Intelligence* 6, Edinburgh University Press, pp. 403-420, 1969