The TRACK Program Package

VISION FLASH 49

by

Jerome B. Lerman
Robert J. Woodham

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Robotics Section

August 1973

Abstract

A collection of LISP functions has been written to provide vidissector users with the following three line-oriented vision primitives:

(i) given an initial point and an estimated initial direction, track a line in that direction until the line terminates.

(ii) given two points, verify the existence of a line joining those two points.

(iii) given the location of a vertex, find suspect directions for possible lines emanating from that vertex.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-005.

Vision flashes are informal papers intended for internal use.

This memo is located in TJ6-able form on file VIS;VF49 >.

1 INTRODUCTION

A key theme in current vision research has been to implement programs in which each new piece of knowledge that is obtained can be used to guide further processing. Shirai{1} has utilized such an approach in a program for the recognition of polyhedra. In this paper, we present a particular LISP implementation of three primitives basic to the Shirai scheme. These primitives are:

(i) given an initial point and an estimated initial direction, TRACK a line in that direction until the line terminates.

(ii) given two points, VERIFY the existence of a line joining those two points.

(iii) given the location of a vertex, FIND-SUSPECT directions for possible lines emanating from that vertex.

In Section 2, we present a brief introduction to the treatment of feature points and line types in the TRACK program package. This section is included in order to provide the necessary understanding of the mechanisms underlying the implementation of the above three primitives.

Sections 3, 4 and 5 form the core of this document. Here we present, respectively, detailed discussions of the TRACK-LINE, VERIFY and FIND-SUSPECTS functions.

In Section 6, we summarize the various debugging and utility features available to users of the TRACK package. In addition, we tabulate, for ease of reference, the various switches and special variables used in the TRACK program.

The TRACK-LINE, VERIFY and FIND-SUSPECTS primitives are designed to facilite interaction between high level BLOCKSWORLD programs and the actual vidissector data. The TRACK package is currently used by WIZARD and CWIZ.

The TRACK program is available as AI: TRACK > JBL; A super-winning FASTLOAD version (making use of NLISP) will shortly be available as AI: TRACK FASL JBL;

Any comments or criticisms concerning the TRACK program package should be addressed to JBL or RWOOD. We would appreciate hearing any and all such suggestions.

2 FEATURE POINTS AND LINE TYPES

As a first order approximation, one might consider the process of line finding to consist of two distinct subproblems:

(i) The detection of feature points<Note 1>.

(ii) The amalgamation of feature points into line segments.

A major goal of the TRACK program package has been to exploit interaction between (i) and (ii) above. Feature points already discovered in a scene strongly constrain where one would expect to find line segments. Similarly, the line segments already discovered in a scene strongly constrain where one would expect to find additional feature points.

We begin in this section, however, by restricting ourselves to the problem of the detection of feature points. In particular, we attempt to answer the following two questions:

How can feature points be characterized?

Based on this characterization, how can feature points be recognized?

2.1 Characterizing Feature Points

A feature point is by definition a localized property of the image intensity array. Rather than consider the various 2-dimensional detection predicates one might apply to image points, we simplify the problem by choosing to characterize feature points according to a profile of intensity values along a linear band perpendicular to the direction of the line for which this point is a feature point.

Not all real edges in a scene have similar cross-sectional intensity profiles. Herskovits and Binford{2} classified edges into three types according to the three characteristic intensity profiles depicted in figure 2.1. Step-type edges occur at contrast boundaries between regions of relatively homogeneous intensity. In BLOCKSWORLD scenes, the light source can often induce a varying intensity across a region. Roof-type edges occur at boundaries between regions whose intensity profiles vary almost symmetrically across the boundary. Finally, edge-effects occur at boundaries representing a sharp highlight (or, in the inverse sense, at boundaries representing a sharp 'lowlight' — typically at a crack where one object is stacked upon another).

If the actual intensity profiles were as clean and smooth as in figure 2.1, we could immediately accept the above characterization of feature points and concern ourselves with implementing a simple-minded recognition algorithm based upon that characterization. However, as one might expect, intensity profiles are generally quite noisy.

Herskovits-Binford Intensity Profile Characterizations

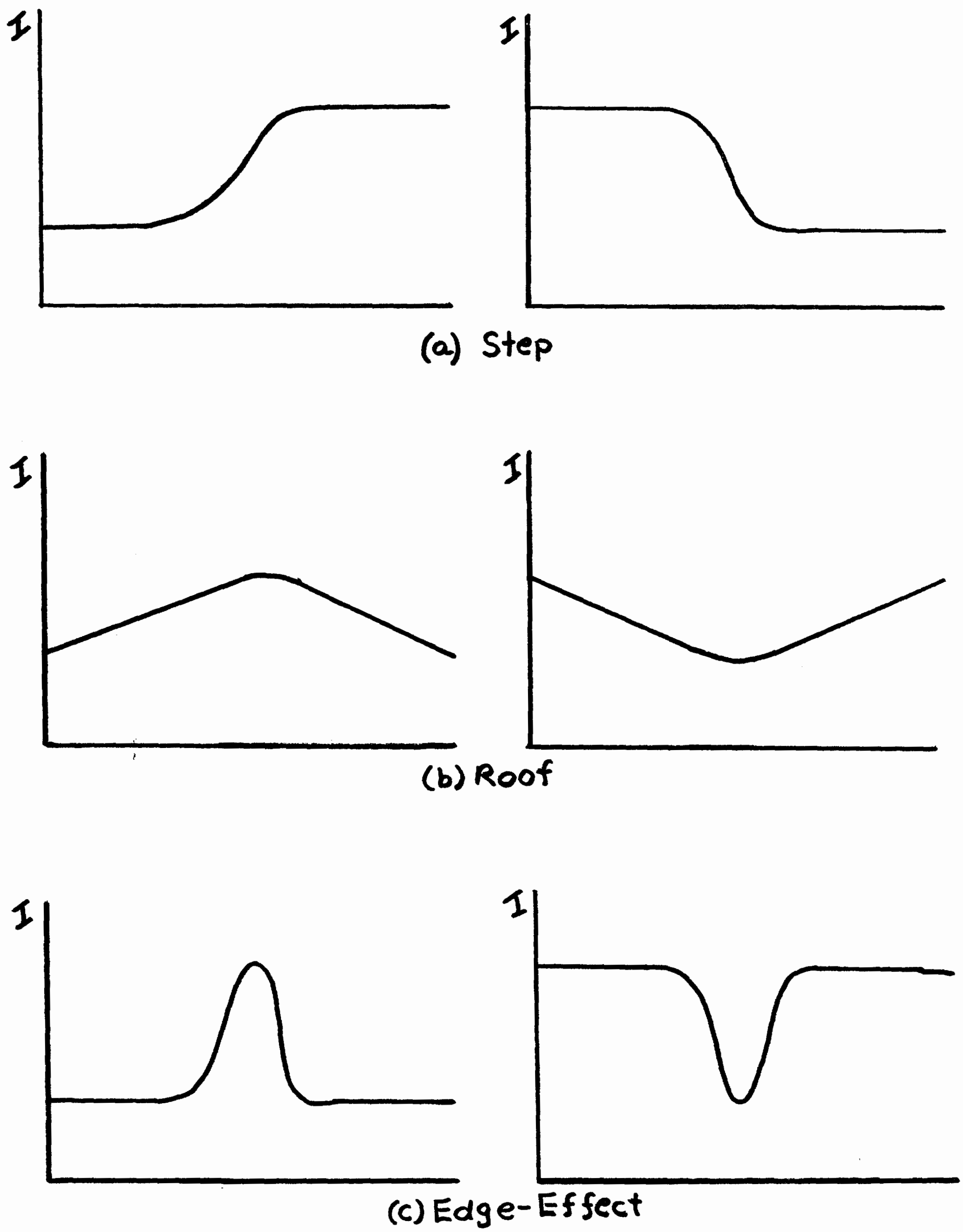


figure 2.1

In our effort to characterize feature points, we seek some filter (predicate) to apply to the actual intensity profile which has the following attributes:

(i) the variance in filter values due to noise is minimized.

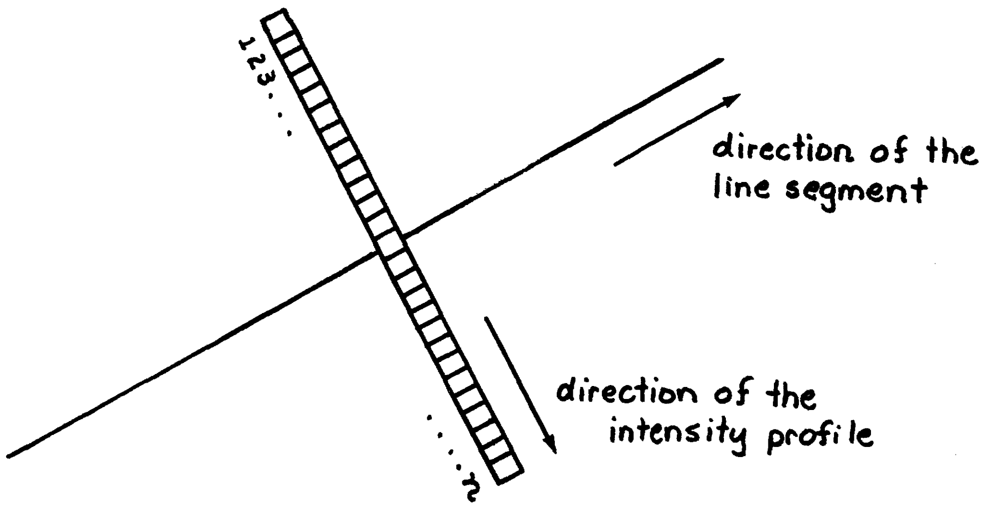
(ii) the variance in filter values due to the actual edge is maximized.

(iii) the filter (predicate) is computationally simple to apply.

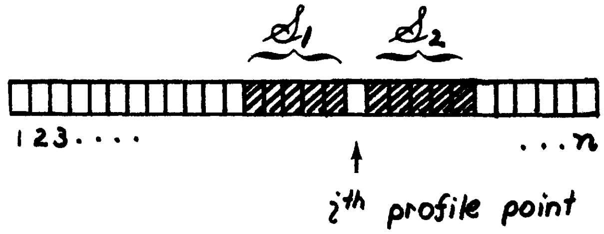
With such a filter, we could base our characterization and subsequent recognition procedure upon a profile of filter values rather than upon a profile of noisy intensity values.

Shirai has defined a contrast function for intensity profiles which is remarkably successful according to the above three criteria. The reader is referred to the Shirai paper{1} for a more detailed discussion. The presentation here is simply in terms of the implementation given Shirai's contrast function in the TRACK program package.

The Shirai filter is conceptually quite simple. Consider an intensity profile of n points taken along a band perpendicular to the direction of the line as shown in figure 2.2(a)<Note 2>. We define the filter-value of the ith profile point, as in figure 2.2(b), to be the difference between the sum of the m subsequent intensity points and the sum of the m preceding intensity points, where m is some parameter. (Thus, there are (n-2m) points for which the contrast function is well defined.)



(a) taking an intensity profile



$$\mathcal{S}_1 = \sum_{r=i-m}^{i-1} I_r \qquad \mathcal{S}_2 = \sum_{r=i+1}^{i+m} I_r$$

the Shirai contrast function at the i^{th} profile point is $F_m(i)$ where

$$F_m(i) = \mathcal{S}_2 - \mathcal{S}_1$$

(b) defining the Shirai contrast function

figure 2.2

In figure 2.3, we show the filter-value profiles corresponding to the various edge-types depicted in figure 2.1. For the purposes of the TRACK program, it has not been necessary to differentiate between edge-effects and roof-type edges. The same detection process works well for both types of lines. Thus, in the TRACK program package, we characterize a feature point into one of four classes:

(i) a MAXIMUM is a feature point representing a step-type edge where the intensity profile crosses from a region of relative brightness to a region of relative darkness<Note 3>.

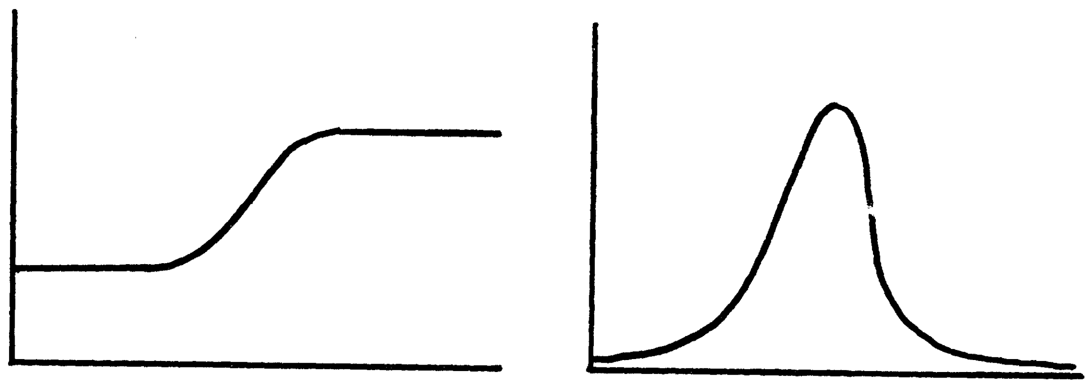
(ii) a MINIMUM is a feature point representing a step-type edge where the intensity profile crosses from a region of relative darkness to a region of relative brightness.

(iii) a HILITE is a feature point representing an edge-effect (or a roof-type edge) where the intensity profile crosses a highlight (ie. a boundary of relative brightness).

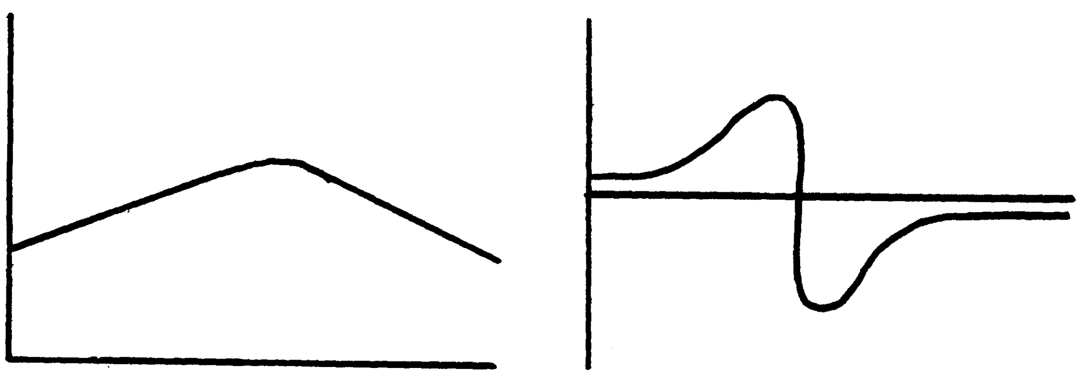
(iv) a CRACK is a feature point representing an edge-effect (or a roof-type edge) where the intensity profile crosses a "lowlight" (ie. a boundary of relative darkness).

We characterize a particular line segment according to the feature points which are found to lie on that line segment. Thus, a line segment is of type MAXIMUM, MINIMUM, HILITE or CRACK respectively as its feature points are of type MAXIMUM, MINIMUM, HILITE or CRACK. We note in passing that reversing the direction in which a particular line segment is tracked causes a MAXIMUM line segment to become a MINIMUM (and vice-versa). However, a line segment will be classified as a

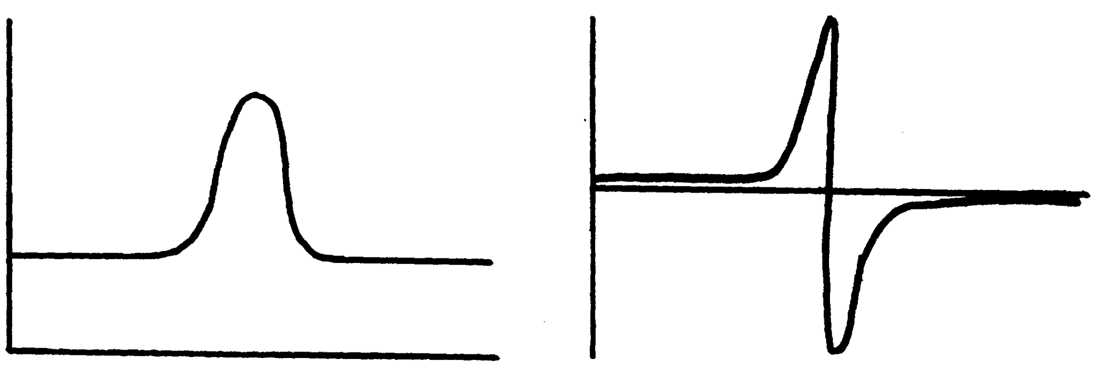
Shirai Filter-Value Profiles



(a) Step



(b) Roof



(c) Edge-Effect

figure 2.3

HILITE or CRACK independent of the direction of the track.

In figures 2.4 and 2.5 we show the intensity profiles<Note 4> and filter-value profiles across two edges from the stored picture POINTS CUBE VIS;.

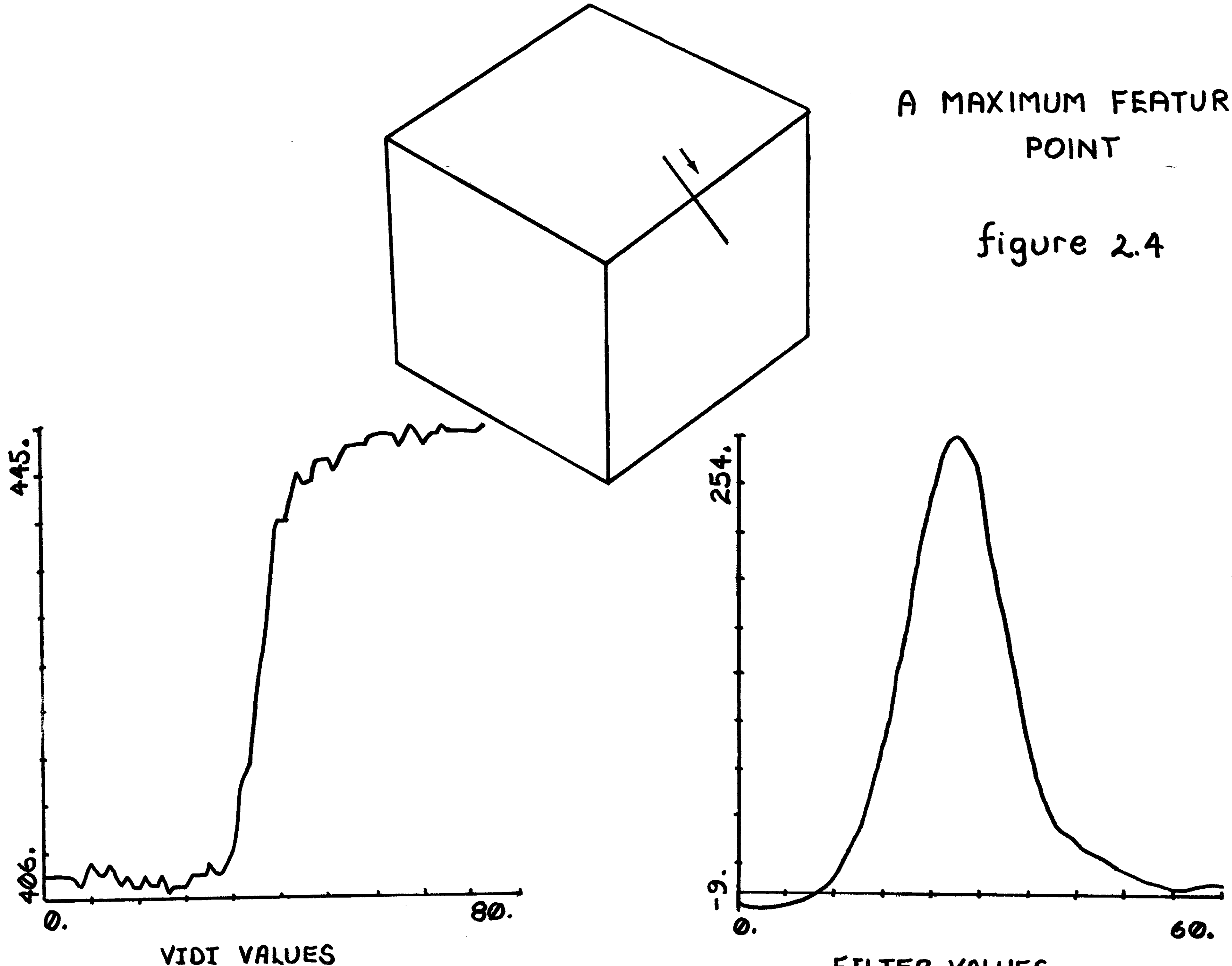
There are two special variables in the TRACK program package which serve as controlling parameters for the generation of intensity and filter-value profiles. The variable DELTA is SETQ'd to the size of the set of support for the contrast function calculation (ie. to m in figure 2.2(b)). The variable PHW (Predicate Half Width) is SETQ'd to the number of filter-values with which to frame each side of the suspected feature point. Thus, the filter-values profile will be of length (1+ (+ PHW PHW)). In order to define this number of filter-values, the intensity profile must be of length (1+ (* 2 (+ PHW DELTA))).

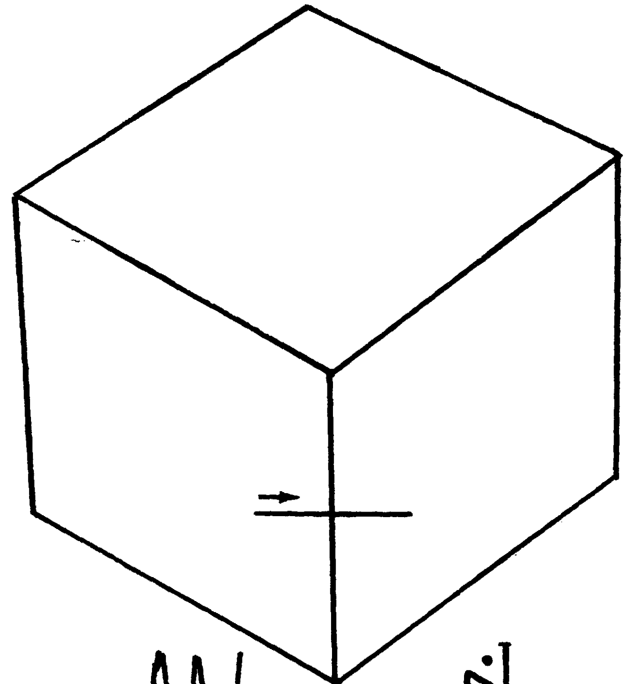
Figure 2.6 shows the filter-value profiles of an edge for various settings of DELTA (throughout this example, PHW was SETQ'd to 30.). A general rule of thumb is that as the value of DELTA grows larger, the effect of noise is reduced while the height of peaks due to edges is increased. However, the width of these peaks also increases so that, in some sense, the contrast function becomes more costly to compute (since the intensity profiles must be taken over a larger area in order to capture the entire width of the edge peaks).

The standard defaults for the TRACK package are such

A MAXIMUM FEATURE POINT

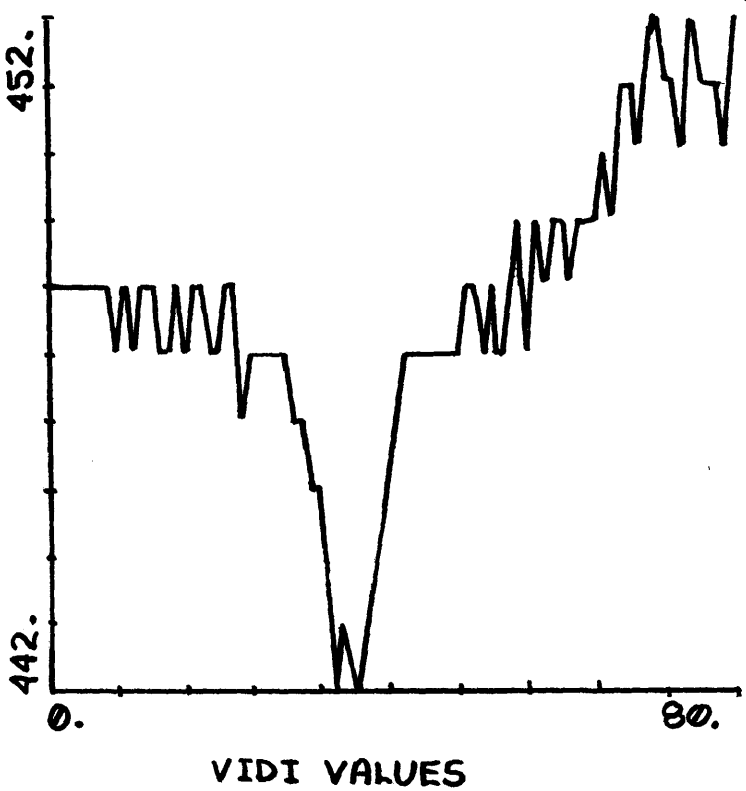
figure 2.4

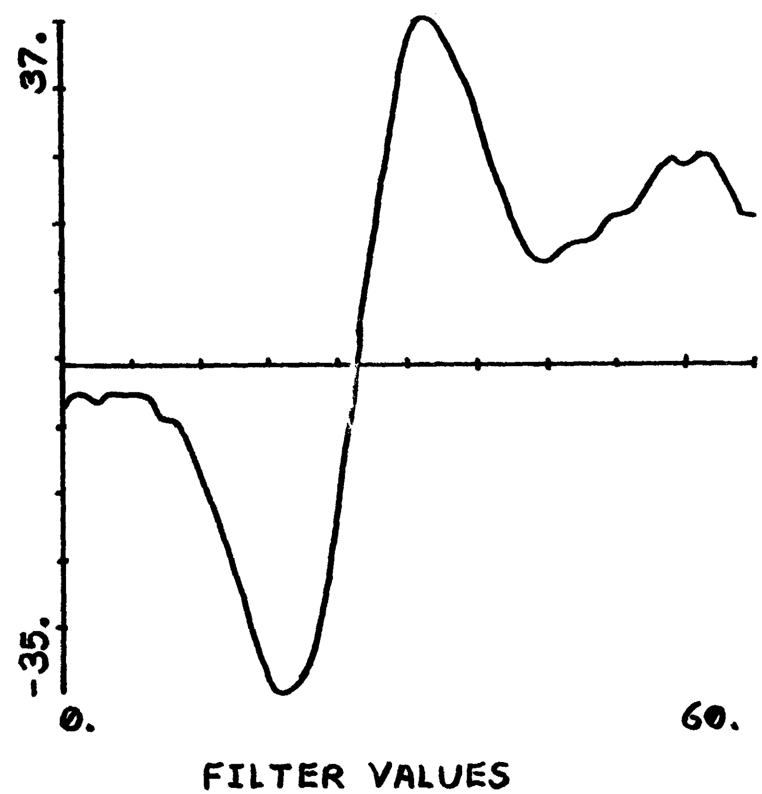




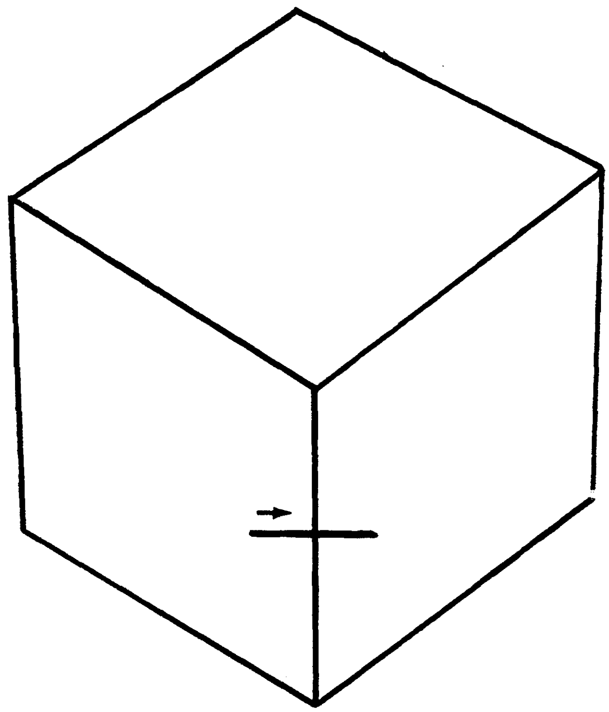
A HILITE FEATURE POINT

figure 2.5

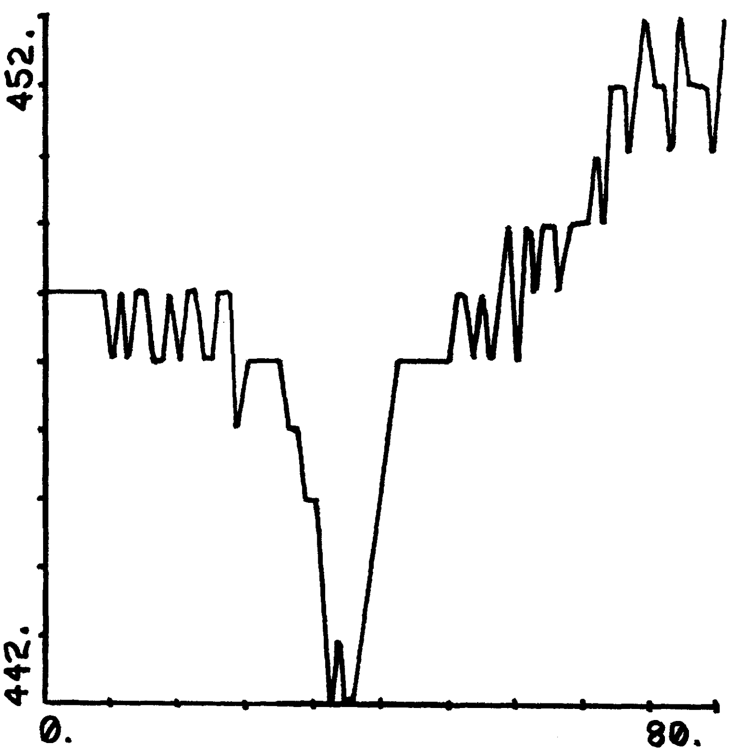




that DELTA=10. and PHW=20. However, if the TRACK program is tracking either a MAXIMUM or a MINIMUM type line, it will temporarily bind PHW to be one half its standard value. (In the case of a MAXIMUM or a MINIMUM, the filter-values profile has to capture only one peak while in the case of a HILITE or CRACK, it must capture two.) Users of the TRACK program package should feel free to experiment with various values of DELTA and PHW.

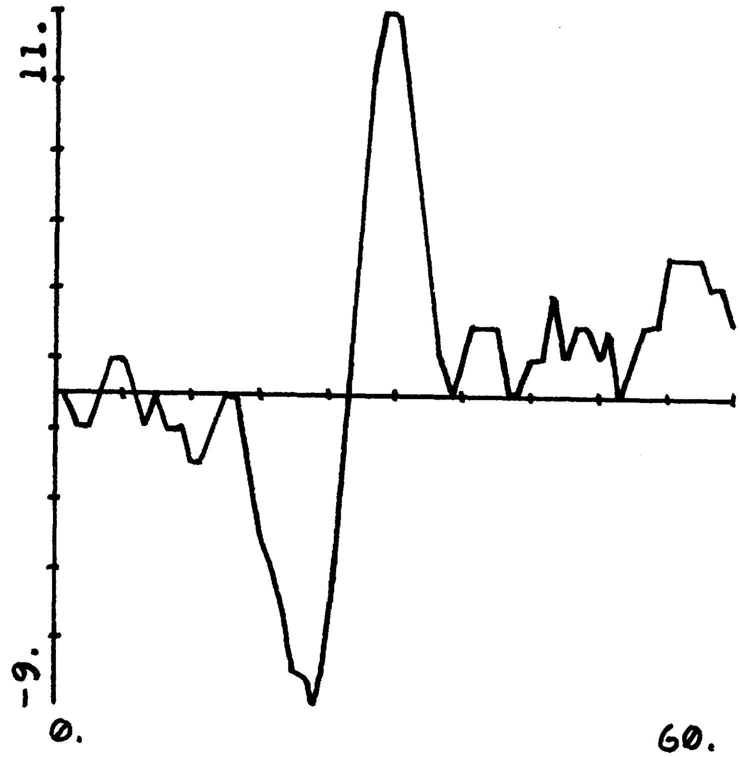


(a) THE SCENE

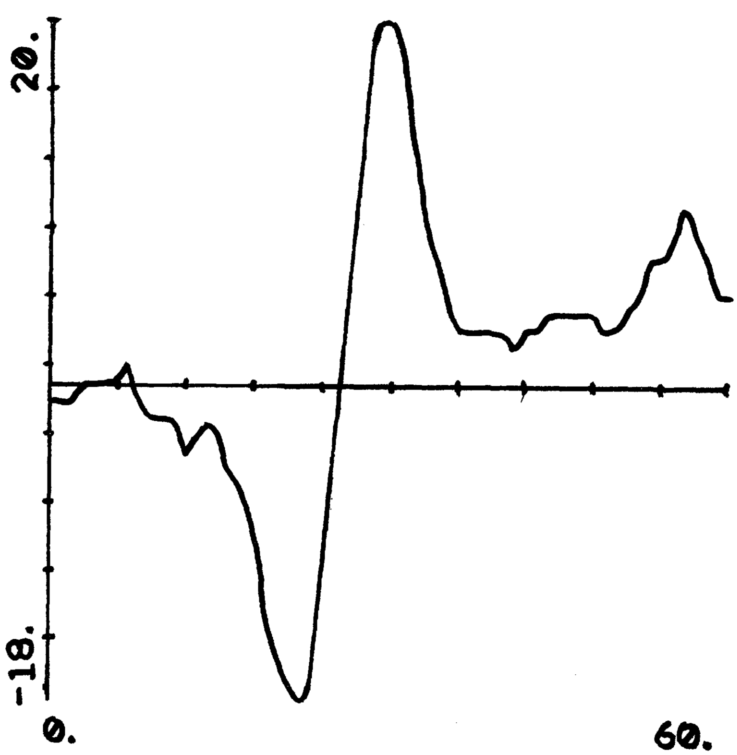


(b) VIDI VALUES PROFILE

figure 2.6 (...CONT'D)

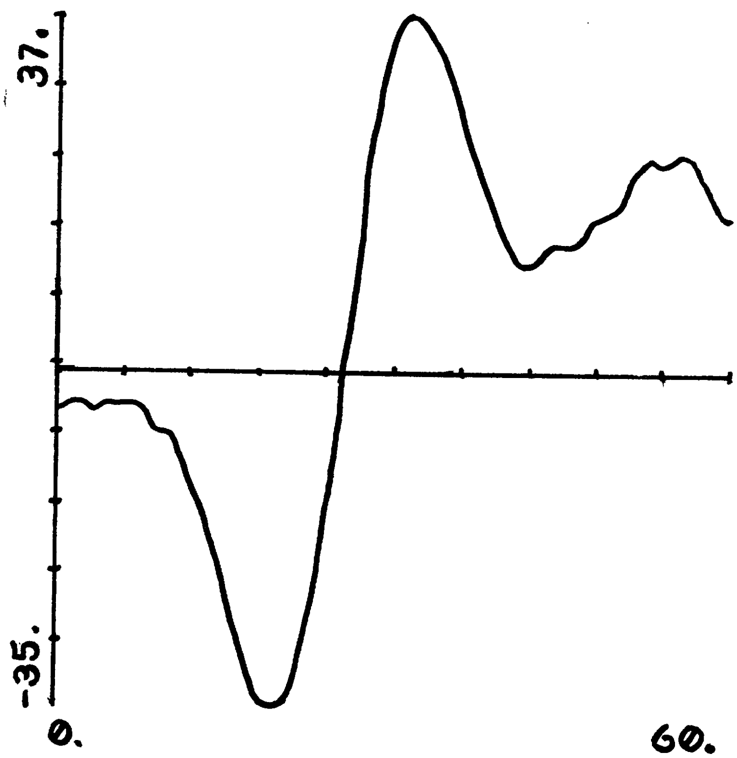


(c) FILTER VALUES DELTA = 3.

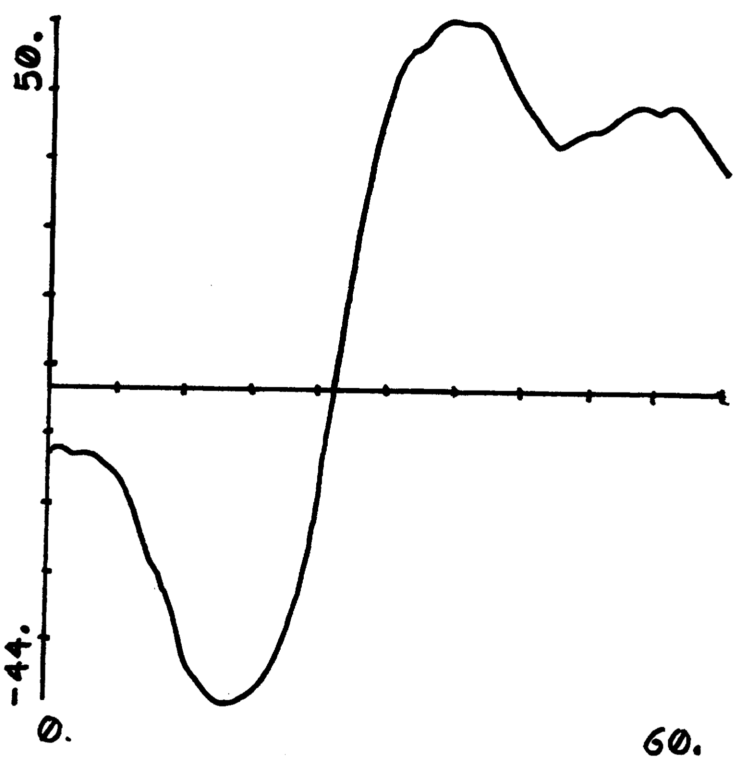


(d) FILTER VALUES DELTA=5.

(...CONT'D)



(e) FILTER VALUES DELTA = 10.



(f) FILTER VALUES DELTA = 15.

2.2 Recognizing Feature Points

In section 2.1 we presented a method for characterizing feature points in a scene. Applying the Shirai contrast function to an intensity profile gives us a filter-values profile which is quite free from noise and whose shape is characteristic of the type of feature point being considered. However, we are still faced with the questions,

> Given an arbitrary filter-values profile, how do we decide whether or not it represents the profile of a feature point?
>
> Given that a filter-values profile represents that of a feature point, how do we decide the type of the feature point and how do we locate the feature point in the profile?

In this section we present the algorithm used in the TRACK program package to analyze the filter-value profiles.

The general algorithm for finding peaks in the filter-values profile is summarized as follows:

(1) Beginning at the leftmost end of the filter-values profile, move to the right until we find an entry whose absolute value is m_1 units greater than the minimum absolute value encountered along the way. Call the position of this entry START and call its value STARTV. If we reach the rightmost end of the filter-values profile before finding such a START, return NIL indicating that there is no feature point along this profile.

(2) Beginning at the rightmost end of the filter-values profile, move to the left until we find an entry whose absolute value is m_1 units greater than the minimum absolute value encountered along the way. Call the position of this entry END and call its value ENDV. If we reach START before finding such an

END, return NIL to indicate that there is no feature point along this profile.

(3) Find the maximum and minimum filter-values between START and END. Call these values respectively MAXV and MINV.

(4) If MAXV < m_2, reject the possibility of there being a MAXIMUM type feature point in this intensity profile.

If both STARTV and ENDV are not less than 90% of MAXV, reject the possibility of there being a MAXIMUM type feature point in this intensity profile.

Otherwise, consider all portions of the filter-values profile between START and END lying above a height 80% of MAXV. Call the midpoint of that portion closest to the line being tracked a MAXIMUM type feature point.

(5) Similarly, if MINV > m_2, reject the possibility of there being a MINIMUM type feature point in this intensity profile.

If both STARTV and ENDV are not greater than 90% of MINV, reject the possibility of there being a MINIMUM type feature point in this intensity profile.

Otherwise, consider all portions of the filter-values profile lying below a height 80% of MINV. Call the midpoint of that portion closest to the line being tracked a MINIMUM type feature point.

(6) If both MAXIMUM and MINIMUM type feature points have been found in (4) and (5) above, and, if the difference in the height of the two respective peaks is no greater than 75% of the largest peak, call the feature point a HILITE or CRACK according to whether we have crossed a MINIMUM-MAXIMUM pair or a MAXIMUM-MINIMUM pair. Otherwise, call the feature point a MAXIMUM or MINIMUM according to which of the MAXIMUM or MINIMUM peaks is closest to the line being tracked.

In figure 2.7 we show a typical filter-values profile analysis. Steps (1) and (2) above are designed to frame the filter-values profile so that the values located between START and END actually represent peaks and gradients due to edge effects. We can think of START and END as defining the

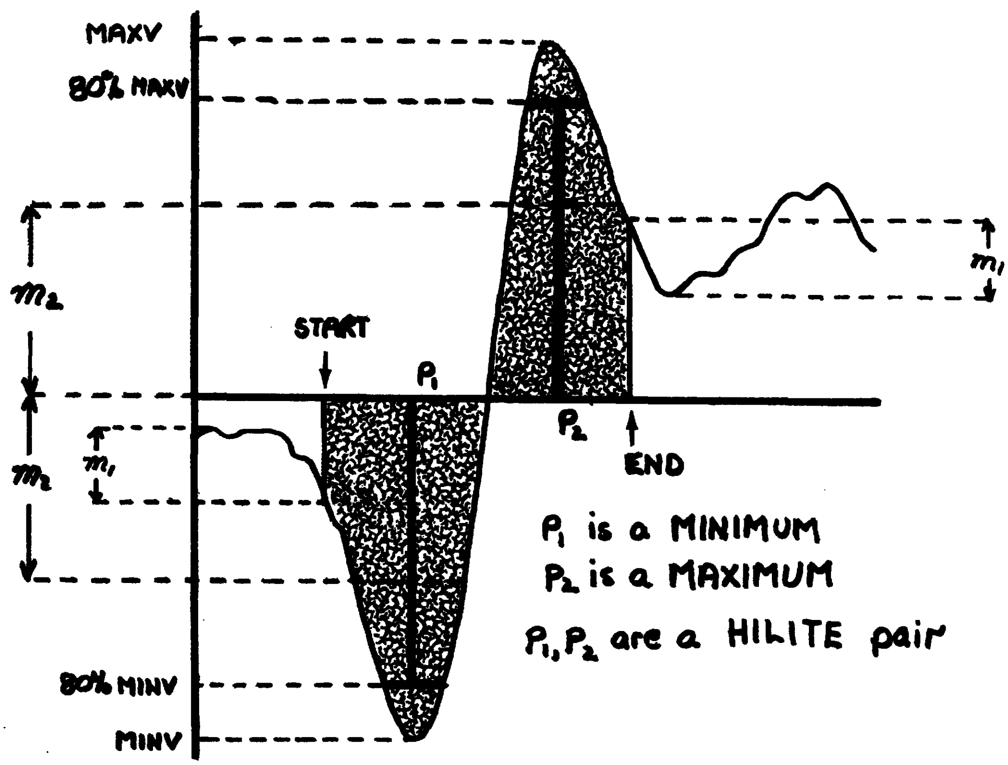


figure 2.7

'shoulders' of the peak. In steps (4) and (5), we see that a particular intensity profile can be rejected as representing that of a feature point either because its MAXV/MINV values are not large enough or because the shoulders of that portion of the peak we have captured in our profile are not low enough. (The parameters m_1 and m_2 are estimated from the height of the peaks already encountered in tracking the current line segment.)

The algorithm above is given for the most general case. It is greatly simplified within TRACK when one already knows the type of feature point being sought. When we know before taking a profile that we are looking for MINIMUM or CRACK type feature points, we reverse the sense of the contrast function (ie. instead of subtracting the sum of the DELTA preceding intensity points from the sum of the DELTA subsequent intensity points, we subtract the sum of the DELTA subsequent from the DELTA preceding.) In this way, we typically need to deal only with the MAXIMUM and MINIMUM-MAXIMUM pair type profiles shown in figures 2.4 and 2.5.

3 TRACKING A LINE

The LEXPR TRACK-LINE is defined such that, given an initial point and an estimated initial direction, it will track a line in that direction until the line terminates. TRACK-LINE is called by:

(TRACK-LINE X Y DIRN TYPE)

where

X is the FIXNUM x-coordinate of the initial point for the track (0. ≤ X < 1024.)

Y is the FIXNUM y-coordinate of the initial point for the track (0. ≤ Y < 1024.)

DIRN is the direction of the line in degrees (FIXNUM or FLONUM) where direction is interpretted to be the directed angle the line makes with respect to the positive x-axis (0. ≤ DIRN < 2π)

TYPE is an optional argument which, if specified, is one of 'MAXIMUM, 'MINIMUM, 'HILITE or 'CRACK indicating the specific type of line to be tracked. (If no TYPE argument is specified, TRACK-LINE will track any type of line it encounters.)

TRACK-LINE returns:

NIL if no line was found (or if no line of the specified type was found).

((x-end-pt y-end-pt) EQUN TYPE)
where x-end-pt and y-end-pt are respectively the x and y coordinates (FIXNUM's in range 0. -> 1024.) of the end-point of the line segment detected, where EQUN is the three element list of FLONUM's (SIN(θ) -COS(θ) ρ) representing the equation of the line segment detected<Note 5> and, finally, where TYPE is one of 'MAXIMUM, 'MINIMUM, 'HILITE or 'CRACK indicating the type of the line segment detected.

The TRACK-LINE function consists of three separate parts:

(i) begin a track in the estimated direction to determine if there is, in fact, a line segment to be tracked. At this stage we determine:
   a) the line type (if it is not already specified)
   b) an updated estimate of the line's direction
   c) an estimate of the average peak height to be expected in the filter-value profiles across this line.

(ii) follow the line segment until the track is lost.

(iii) find, as accurately as possible, the end-point of the line segment.

3.1 Beginning a Track

The beginning of a track is handled slightly differently depending upon whether or not a line type is specified in the call to TRACK-LINE.

3.1.1 Beginning a Track of Unknown Type

In beginning a track of unknown type, TRACK-LINE examines 5. intensity profiles across points spaced between 15. and 30. vidissector units along the estimated path of the line segment<Note 6>. The analysis of these 5. profiles proceeds as described in section 2.2. However, rather than attempting to decide a line type associated with each profile, the decision is postponed until all 5. profiles have been examined.

All the MAXIMUM feature points encountered are entered onto a list called MAXLIST. Similarly, the MINIMUM feature points are entered onto a list called MINLIST. In addition, all the MAXIMUM-MINIMUM pairs that qualify as a CRACK are entered onto a list called CRACKLIST. Finally, MINUMUM-MAXIMUM pairs that qualify as a HILITE are entered onto a list called HILIST.

If either CRACKLIST or HILIST has 3. or more entries, the line type is said, respectively, to be a CRACK or a HILITE. Otherwise, if either MAXLIST or MINLIST has 3. or more entries, the line type is said, respectively, to be a MAXIMUM or a MINIMUM. Finally, if none of the feature point lists has 3. or more entries, TRACK-LINE returns NIL indicating that no line segment has been found.

TRACK-LINE now knows the type of the line segment to be tracked. In addition, the position of the feature points encountered provides an updated estimate of the direction of the line segment while the average peak height of the feature points encountered allows us to "tune" the feature point detection process for the particular line segment in question. (ie. to adjust the values of m_1 and m_2 in the algorithm of section 2.2)

3.1.2 Beginning a Track of Known Type

If the user has already specified the type of the line segment, TRACK-LINE does not have to be quite as laborious at the beginning of a track. However, rather than immediately begin following the line segment, we still need to update our estimate of the line segment direction and to determine an estimate of the average peak height to be expected.

To do this, TRACK-LINE examines 3. intensity profiles across points spaced between 15. and 30. vidissector units along the estimated path of the line segment. These 3. profiles are analyzed specifically for feature points of the type in question. If fewer than 2. of the profiles show feature points of the required type, TRACK-LINE returns NIL indicating that no line of the specified type has been found. Otherwise, TRACK-LINE uses the feature points encountered to update its estimate of the line segment direction and to "tune" the detection process as in section 3.1.1.

3.2 Following a Line Segment

Following a line segment can be seen as consisting of two basic operations:

(i) Predicting where a feature point should lie along the line segment.

(ii) Determining whether or not there is an appropriate feature point sufficiently close to the predicted location.

The prediction operation for (i) above is simply stated. Prediction consists of extrapolating the line segment some distance beyond its current end-point using a least-mean-square equation of the line as determined from the feature points already found to lie on the line.

For (ii) above, we again will be applying the feature point recognition algorithm of section 2.2. However, at this point, we are left with a certain vagueness to the notions of "appropriate" and "sufficiently close" stated above. Most of the 'hair' associated with the line following algorithm given below is concerned with making precise just what we mean by "appropriate" and "sufficiently close".

Before detailing the algorithm, we begin with a brief discussion of the parameters involved. Suppose that d is the distance in the filter-values profile between the actual feature point and its predicted location. Two special variables D1 and D2 determine our interpretation of the feature point. If $d \leqslant D1$ then we say the feature point is a GOOD POINT (ie. the point definitely lies on the line). If $d > D2$ then we say the feature point does not lie on the line. If $D1 < d \leqslant D2$ then we say the feature point is an AMBIGUOUS POINT and we defer a decision until subsequent analysis. D2 is constant throughout and its default is $D2 = 8.5$. D1 varies during the line following algorithm but it is initially SETQ'd to the value of DO (whose default is $DO = 3.0$).

A number of parameters are associated with terminating conditions for the line following algorithm. The special variables M1 and M2 are used to record the current status of line following. One can think of M1 as counting the number of times we have failed to find a feature point on the line since the last GOOD POINT and of M2 as counting the number of AMBIGUOUS POINT's since the last GOOD POINT. However, the above is only a first approximation since, in fact, the interpretation of M1 and M2 is affected by special variable MA (default value is MA = 1.).

The variable MA determines how willing we are to accept AMBIGUOUS POINT's as GOOD POINT's after having failed, at some previous stage, to find any feature point at all on the line. (Refer to the algorithm below for a precise interpretation.) The follow algorithm will terminate if either M1 > MN or (+ M1 M2) > MT. (The defaults for MN and MT are MN = 3 and MT = 7).

Finally, two special variables serve as parameters for the prediction operation. STEP (default value is STEP = 10.0) is the basic step size in vidissector units used for extrapolating beyond the current feature point. M is an 'enthusiasm factor' used to scale the basic step size according to the number of successive GOOD POINT's we have previously encountered. (M is SETQ'd to (+$ 1.0 (*$ 0.5 n)) where n is the number of successive GOOD POINTS). The actual step size used for prediction is (*$ M STEP).

We are now ready to summarize the line following algorthm:

(1) Initialize for following a line by SETQ'ing M = 1.0, M1 = 0., M2 = 0. and D1 = DO. Also, initialize XY to be the last feature point found by begin-track and EQ to be the updated equation of the line as determined by begin-track.

(2) SETQ XY to be the point (*$ M STEP) units past the current XY along the line given by EQ. Obtain an intensity profile centered at XY.

(3) If no feature point of the required type is found, SETQ M1 to (1+ M1) and M to 1.0
Go to (7).

(4) If d ≤ D1, consider the current XY to be a GOOD POINT. If M1 > MA SETQ M1 to (1- M1). Otherwise, consider as GOOD POINT's all feature points previously considered to be AMBIGUOUS POINT's and SETQ M1 to 0., M2 to 0. and D1 to DO. Update EQ to be the least-mean-square equation based upon all feature points now considered as GOOD POINT's<Note 7>. SETQ M to (+$ M 0.5).
Go to (7)

(5) If D1 < d ≤ D2, consider the current XY to be an AMBIGUOUS POINT. SETQ M to 1.0, M2 to (1+ M2) and D1 to (+$ DO (*$ M2 WM))<Note 8>. If M1 > MA, SETQ M1 to (1- M1).
Go to (7)

(6) If d > D2, consider the point to be off the line. SETQ M to 1.0 and M1 (1+ M1).

(7) If M1 ≤ MN or if (+ M1 M2) ≤ MT then go to (2). Otherwise, consider the line segment to be lost and terminate the follow procedure.

3.3 Finding the End-point of the Line Segment

When TRACK-LINE loses a line segment, we know only that the line segment must have terminated somewhere between the last GOOD POINT obtained and the point examined immediately following that point<Note 9>. We still need to find, as accurately as possible, the end-point of the line segment.

TRACK-LINE finds the end-point of a line segment by employing a simple binary search algorithm. An intensity profile is taken at a point midway between the last GOOD POINT obtained and the subsequent point tried. If that profile reveals a GOOD POINT, its position becomes that of the last GOOD POINT obtained. Otherwise, its position becomes that of the subsequent point tried. We iterate this procedure until the distance separating the last GOOD POINT obtained and the subsequent point tried is not more than 2. vidissector units<Note 10>.

In figure 3.1 we show an example of TRACK-LINE applied to a line in the stored picture POINTS CUBE VIS;

TRACKING A LINE SEGMENT

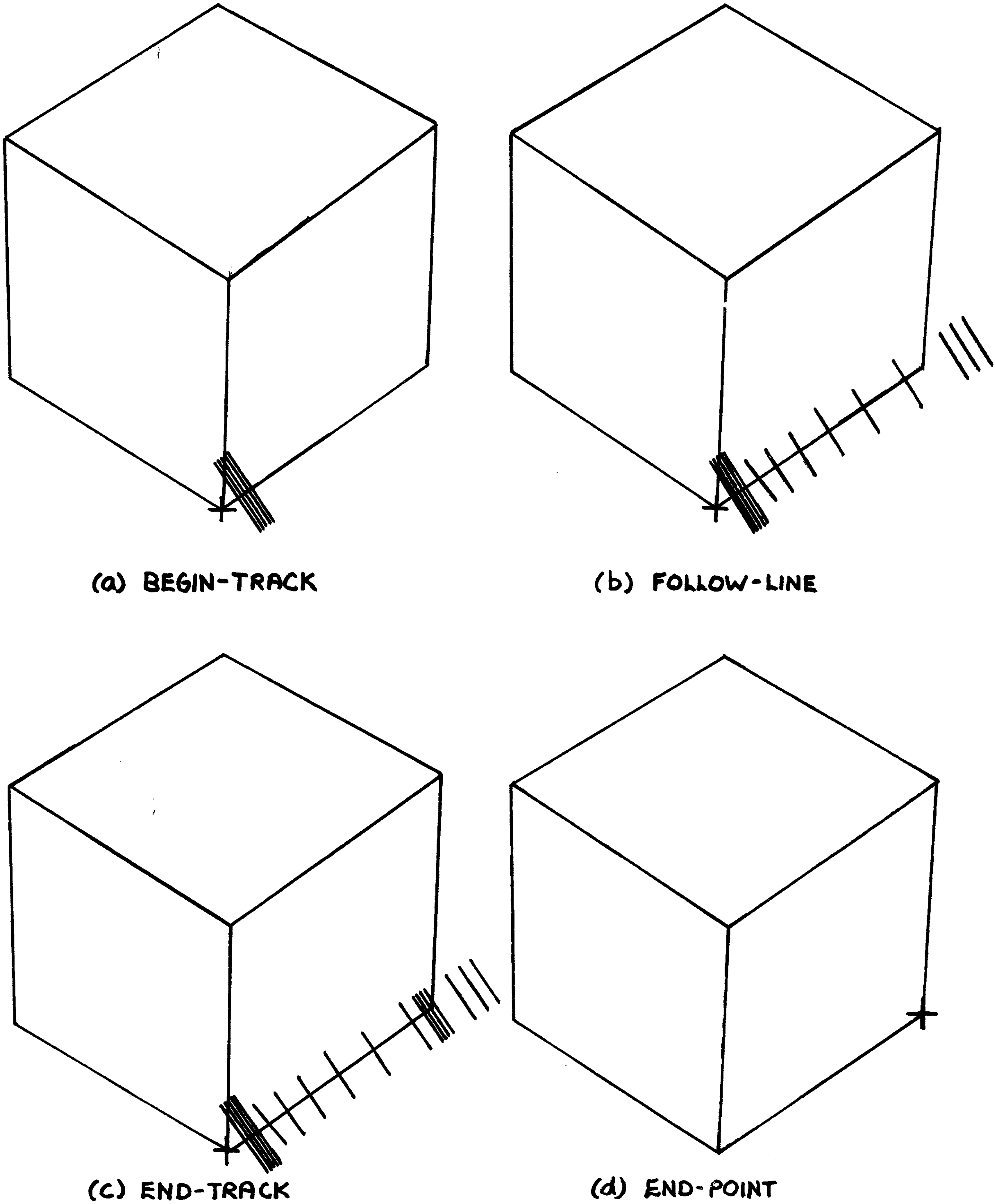


figure 3.1

4 VERIFYING A LINE

The LEXPR VERIFY is defined such that, given two points, it will verify the existence of a line joining those two points. VERIFY is called by:

(VERIFY X1 Y1 X2 Y2 TYPE NUM)

where

X1 is the FIXNUM x-coordinate of the first end-point of the line segment to be verified. ($0. \leq X1 < 1024.$)

Y1 is the FIXNUM y-coordinate of the first end-point of the line segment to be verified. ($0. \leq Y1 < 1024.$)

X2 is the FIXNUM x-coordinate of the second end-point of the line segment to be verified. ($0. \leq X2 < 1024.$)

Y2 is the FIXNUM y-coordinate of the second end-point of the line segment to be verified. ($0. \leq Y2 < 1024.$)

TYPE is an optional fifth argument which, if specified, is one of 'MAXIMUM, 'MINIMUM, 'HILITE, 'CRACK or 'ALL indicating the specific type of line to be verified. (If no fifth argument is specified or if TYPE is 'ALL, VERIFY will verify any type of line occuring between (X1,Y1) and (X2,Y2).)

NUM is an optional sixth argument which, if specified, is the FIXNUM number of intensity profiles VERIFY will use in attempting to verify the line segment. (If no sixth argument is specified, VERIFY will use 5. intensity profiles.)

VERIFY returns:

NIL if less than 80% of the intensity profiles tested were found to have feature points (or if less than 80% of the intensity profiles tested were found to have feature points of the specified type).

(TYPE EQUN)
where TYPE is one of 'MAXIMUM, 'MINIMUM, 'HILITE or

'CRACK indicating the type of the line segment verified and where EQUN is the three element list of FLONUM's ((SIN(θ) -COS(θ) ρ) representing an updated equation of the line verified<Note 5>.

With the recognition algorithm of section 2.2 at hand, verifying a line segment is quite straightforward. VERIFY will examine NUM successive intensity profiles evenly spaced across the line segment from (X1,Y1) to (X2,Y2). If, in examining these NUM profiles, 20% of them fail to have feature points (or fail to have feature points of the required type), VERIFY will immediately return NIL indicating that it could not verify the existence of the specified line segment.

Otherwise, VERIFY accumulates the results of its feature point analysis on the four lists MAXLIST, MINLIST, HILIST and CRACKLIST as was described for TRACK-LINE in section 3.1.1<Note 11>. VERIFY calls the line segment a HILITE or a CRACK respectively as either HILIST or CRACKLIST contains greater than 80% of NUM feature points. Otherwise, VERIFY calls the line segment a MAXIMUM or a MINIMUM according to which of MAXLIST or MINLIST is of greater length.

Finally, VERIFY uses the feature points on MAXLIST, MINLIST, HILIST or CRACKLIST (according to the line type) to calculate an updated equation of the line segment.

In figure 4.1 we show an example of VERIFY applied to a line segment in the stored picture POINTS CUBE VIS;.

VERIFYING A LINE SEGMENT

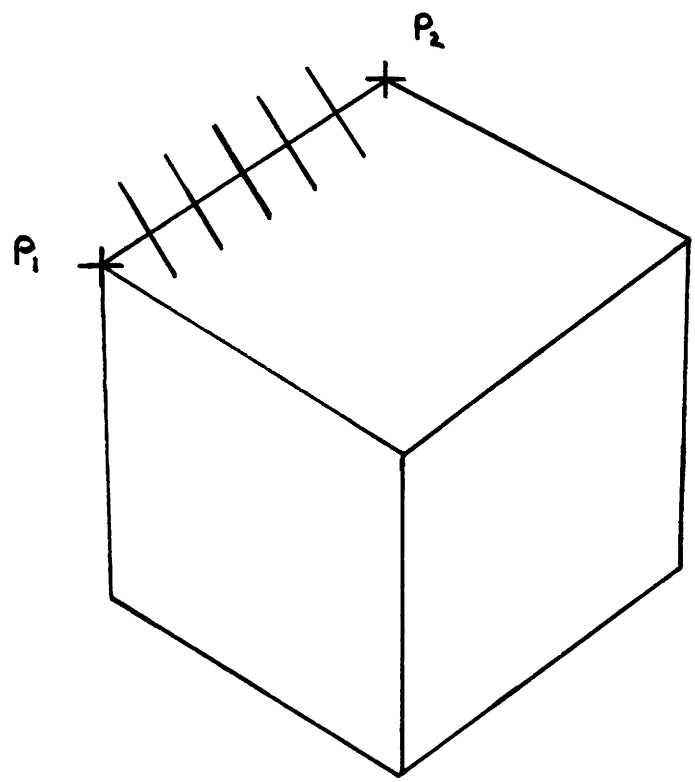


figure 4.1

5 FINDING SUSPECT DIRECTIONS AT A VERTEX

The LEXPR FIND-SUSPECTS is defined such that, given the location of a vertex, it will find suspect directions for possible lines emanating from that vertex. FIND-SUSPECTS is called by:

(FIND-SUSPECTS X Y RAD THRSH)

where

X is the FIXNUM x-coordinate of the vertex ($0. \leqslant X < 1024.$)

Y is the FIXNUM y-coordinate of the vertex ($0. \leqslant Y < 1024.$)

RAD is an optional third argument which, if specified, is the radius in vidissector units (FIXNUM or FLONUM) of the circle to be used for the circular intensity profile — see details below. (If no third argument is specified, RAD will be SETQ'd to its default value RAD = 30.)

THRSH is an optional fourth argument which, if specified, is a FIXNUM threshold used in the analysis of the circular filter-values profile — see details below. (If no fourth argument is specified, THRSH will be SETQ'd to its default value THRSH = 10.)

FIND-SUSPECTS returns:

NIL if no suspect directions could be found for possible lines emanating from this 'vertex'.

((DIRN1 TYPE1) (DIRN2 TYPE2) ... (DIRNn TYPEn)) where each DIRNi is the FLONUM angle of a possible line of type TYPEi emanating from (X,Y). (All angles are in the range $0 \rightarrow 2\Pi$ and are measured counterclockwise with respect to the positive x-axis.) Each TYPEi is one of 'MAXIMUM, 'MINIMUM, 'HILITE or 'CRACK (as described in section 2.1). The DIRNi's are ordered so that

DIRN1 > DIRN2 > ... > DIRNn

In section 2, we discussed the characterization and recognition of the feature points of a line based upon a profile of intensity values taken along a linear band perpendicular to the direction of the line. With only slight modification, we use this same technique to characterize and recognize the feature points of possible lines emanating from a vertex.

Consider a circular profile of intensity values centered at the location of a vertex. This circular profile will cross each line segment emanating from the vertex in a direction which can be considered to be perpendicular (at least perpendicular in some small neighborhood surrounding each line segment).

Using the same contrast function defined in section 2.1, we can calculate the circular filter-values profile corresponding to a circular profile of intensity values. In figure 5.1 we show the VIDI-VALUES<Note 12> and FILTER-VALUES circular profiles taken around the indicated vertex in the stored picture POINTS CUBE VIS;.

Again, we characterize feature points in the circular intensity profiles as MAXIMUM, MINIMUM, HILITE or CRACK according to the criteria of section 2.1. Each such feature point is interpretted as indicating a possible line segment emanating from the point (X,Y).

However, the recognition algorithm of section 2.2 can

FINDING SUSPECTS

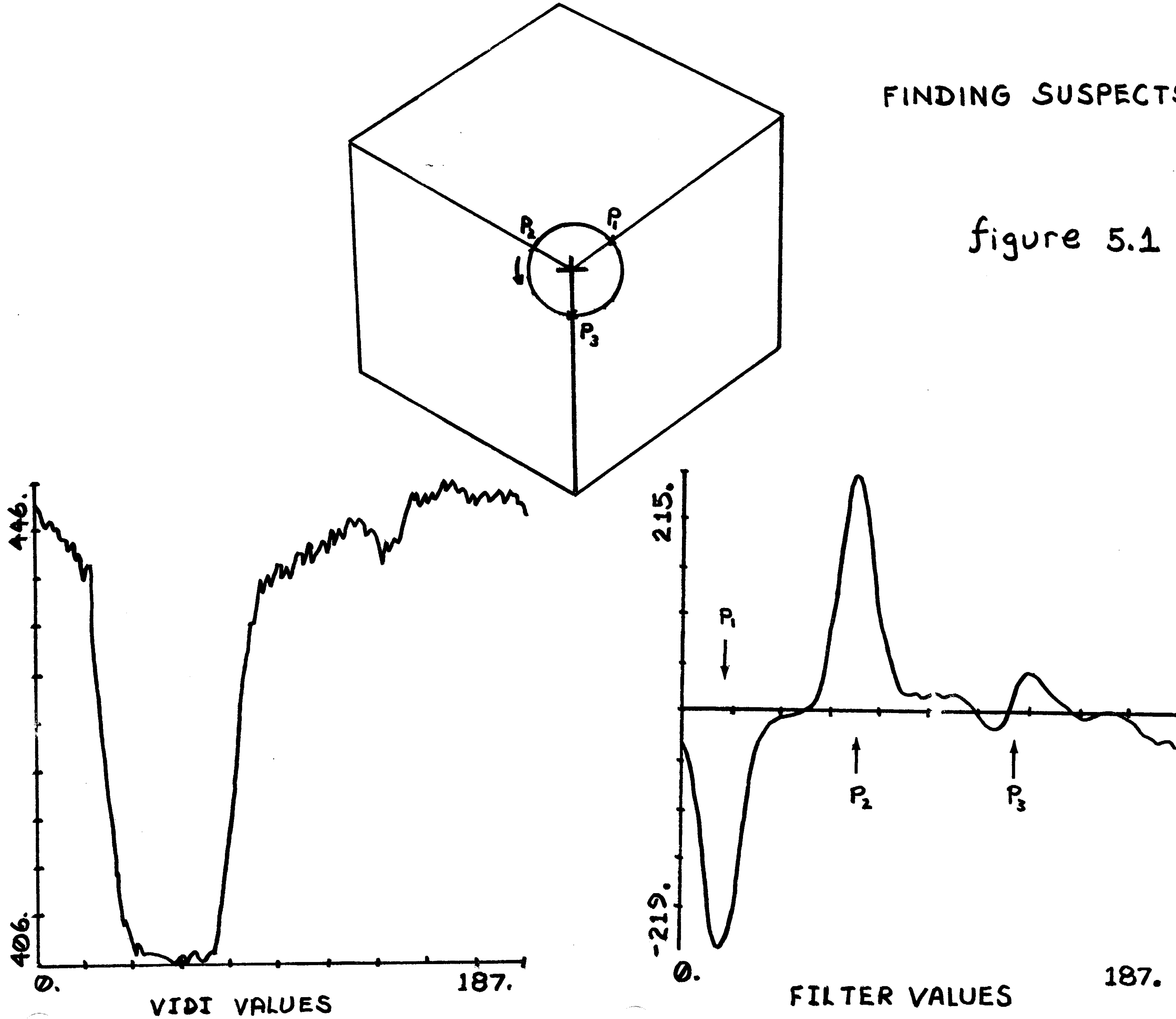


figure 5.1

not immediately be applied to circular filter-value profiles. In section 2.2, we have made the implicit assumption that our filter-values profile has appropriately framed any feature point we might be interested in. In the case of a circular filter-values profile, we have no 'a priori' knowledge of where to expect peaks due to feature points. Nonetheless, the algorithm used by FIND-SUSPECTS to recognize feature points in the filter-values profile is of the same flavor as that of section 2.2. It is summarized as follows:

(1) Beginning at the zero degree heading, move counterclockwise around the filter-values profile recording the type, position and height of all local extrema encountered. Let PLIST be the list of these local extrema of the form

((T1 I1 H1) (T2 I2 H2) ... (Tn In Hn))

where

- (i) T_i is MAXIMUM or MINIMUM<Note 13>.
- (ii) I_i is the index of the extrema in the filter-values profile ($I1 < I2 < \ldots < In$)
- (iii) H_i is the height of the extrema

(2) Flush from consideration as a peak due to a feature point:

- (i) all MAXIMUM type local extrema in PLIST whose height is less than zero.
- (ii) all MINIMUM type local extrema in PLIST whose height is greater than zero.
- (iii) all local extrema in PLIST whose absolute height is less than THRSH.

(3) Consider as a possible peak due to a MAXIMUM feature point any extrema P left in PLIST that is:

- (i) of type MINIMUM<Note 14>.
- (ii) immediately preceeded by a local MAXIMUM whose height is greater than 90% of the height of P.
- (iii) immediately succeeded by a local MAXIMUM whose height is greater than 90% of the height of P.

(4) Consider as a possible peak due to a MINIMUM feature point any extrema P left in PLIST that is:
  (i) of type MAXIMUM<Note 14>.
  (ii) immediately preceeded by a local MINIMUM whose height is less than 90% of the height of P.
  (iii) immediately succeeded by a local MINIMUM whose height is less than 90% of the height of P.

(5) Update the considered position of each possible feature point discovered in (3) and (4) above to be, as in figure 2.7, at the midpoint of the segment joining points taken before and after the extrema at a height 80% of the extrema's height. Convert each position from an index in the filter-values profile to the actual angle of emanation from the point (X,Y).

(6) Consider as HILITE and CRACK feature points, respectively any MINIMUM-MAXIMUM and MAXIMUM-MINIMUM pairs that:
  (i) emanate from (X,Y) at angles differing by no more than
  (//$ (*$ 3.0 (FLOAT DELTA)) RAD)
  radians<Note 15>.
  (ii) are of height whose magnitudes differ by no more than 75% of that of the largest peak.

(7) Return the list
((DIRN1 TYPE1) (DIRN2 TYPE2) ... (DIRNn TYPEn))
for each feature point of a possible line emanating from the point (X,Y) where
DIRN1 > DIRN2 > ... > DIRNn.

6 DEBUGGING AIDS, UTILITY FEATURES, SWITCHES, ETC.

In this final section, we summarize the various debugging and utility features available to users of the TRACK program package.

6.1 Debugging Aids

There are two switches controlling the verbosity of TRACK's communication with the user during TRACK-LINE, VERIFY and FIND-SUSPECTS operations:

DEBUG1 If DEBUG1 ≠ NIL, TRACK will report cursory comments on the user's console concerning the status of TRACK-LINE, VERIFY and FIND-SUSPECTS operations.

DEBUG2 If DEBUG2 ≠ NIL, TRACK will report an in depth analysis on the user's console concerning the status and progress of TRACK-LINE, VERIFY and FIND-SUSPECTS operations.

If both DEBUG1 and DEBUG2 are NIL, TRACK will be silent.

There are two switches which control TRACK's use of the 340 Display:

DISPLAY If DISPLAY ≠ NIL, TRACK will display the progress of all TRACK-LINE, VERIFY and FIND-SUSPECTS operations.

GRAPHF If DISPLAY ≠ NIL, then GRAPHF ≠ NIL will cause each intensity profile and filter-values profile to be graphed respectively in the lower left and lower right corners of the display.

The LEXPR LINE-TEST is a useful debugging function for examining intensity and filter-values profiles across feature points according to the current values of DELTA and PHW. It is called by:

(LINE-TEST X Y DIRN THRSH)

where

X is the FIXNUM x-coordinate of the suspected feature point (0. $\leqslant$ X < 1024.)

Y is the FIXNUM y-coordinate of the suspected feature point (0. $\leqslant$ Y < 1024.)

DIRN is the direction (FIXNUM or FLONUM) of the line for which the point (X,Y) is a suspected feature point, where direction is interpretted to be the angle in degrees that the line makes with respect to the positive x-axis. (0. $\leqslant$ DIRN < 2Π).

THRSH is an optional fourth argument which, if specified, is a FIXNUM estimate of the height of the peak expected in the filter-values profile.(If no fourth argument is specified, THRSH is SETQ'd to its default value THRSH = 10.)

LINE-TEST returns:

NIL if the recognition algorithm of section 2.2 failed to find a feature point using the current values of DELTA, PHW and THRSH.

(TYPE POSN HITE) where TYPE is one of 'MAXIMUM, 'MINIMUM, 'HILITE or 'CRACK indicating the type of the feature point detected and where POSN and HITE are respectively the position and the height of the detected feature point.

Two FEXPR's SHOW and GET-LINES are useful for initializing and controlling the status of the 340 display and the vidissector. SHOW is called by:

(SHOW ARG)

where

ARG = T
implies initialize the display and use the real vidissector for input.

ARG = (FLN1 FLN2 DEV USR)
implies initialize for FTV input using the file

FLN1 FLN2 DEV USR. Also, grab the 340 and display the file IMAGE FLN2 DEV USR, if such a file exists.

ARG = NIL
implies turn off the display.

SHOW returns the item number of the display item created (NIL if no display item was created).

GET-LINES is used to display a line drawing at a given brightness. GET-LINES is called by:

```
(GET-LINES (FLN1 FLN2 DEV USR) N)
```

where

(FLN1 FLN2 DEV USR)
is the filename of the line drawing to be displayed. (GET-LINES is a FEXPR so don't quote the filename.)

N is an optional second argument which, if specified, is the FIXNUM brightness to be used in displaying the line drawing ($1 \leqslant N \leqslant 8.$). (if no second argument is specified, N will be SETQ'd to its default value N = 2.

GET-LINES returns the item number of the display item used.

6.2 Utility Functions

The TRACK program package contains several functions useful for creating and manipulating line equations<Note 5>.

The EXPR L-M-S is defined such that, given a list line equations, it will calculate the least-mean-square error point of intesection of the equations. L-M-S is called by:

```
(L-M-S '(EQ1 EQ2 ... EQn))
```

where

EQ_i is a list of the form (A B C) where the equation of the line is AX + BY + C = 0.

L-M-S returns:

NIL if less than 2. lines were specified

PARALLEL if the lines were parallel.

(X Y)
where X and Y are respectively the FIXNUM x-coordinate and the FIXNUM y-coordinate of the least-mean-square estimate of the point of intersection.

L-M-S is used within TRACK to project a number of line segments to a single vertex. In the case of two equations specified in the call, L-M-S is an exact procedure for finding their point of intersection.

The EXPR MAKELINE is defined such that, given a list of points, it calculates a least-mean-square estimate of the equation of the line passing through these points. MAKELINE is called by:

```
(MAKELINE '((X1 Y1) (X2 Y2) ... (Xn Yn))
```

where

X is the FIXNUM x-coordinate of the ith point

Y is the Fixnum y-coordinate of the ith point.

MAKELINE returns:

(SIN(θ) -COS(θ) ρ)
where the equation of the line is given by $\mathrm{SIN}(\theta)X - \mathrm{COS}(\theta)Y + \rho = 0$.

The EXPR SUM-LIST is called by MAKELINE. SUM-LIST takes a list of points and calculates the statistics required by MAKE-EQUATION. SUM-LIST is called by:

```
(SUM-LIST '((X1 Y1) (X2 Y2) ... (Xn Yn))
```

where each X_i and Y_i are as above.

SUM-LIST returns:

$$(\sum X \quad \sum Y \quad \sum XX \quad \sum YY \quad \sum XY \quad N)$$

The EXPR MAKE-EQUATION takes the sums of X, Y, XX, YY and XY along with N, the number of points, and calculates the line equation. MAKE-EQUATION is called by:

$$(\text{MAKE-EQUATION} \quad \sum X \quad \sum Y \quad \sum XX \quad \sum YY \quad \sum XY \quad N)$$

and returns (SIN(θ) -COS(θ) ρ) as above.

The EXPR EQ-ANGLE takes the equation of a line and calculates the angle that the line makes with respect to the positive x-axis. EQ-ANGLE is called by

(EQ-ANGLE '(A B C))

where the equation of the line is AX + BY + C = 0. EQ-ANGLE returns the angle ANG where $0. \leq ANG < 2\Pi$.

The above functions L-M-S, MAKELINE, SUM-LIST, MAKE-EQUATION and EQ-ANGLE are available indepedently in VISLIB.

6.3 Special Variables

Several special variables serve as parameters for TRACK and may be changed by the user. They are summarized as follows:

DELTA Standard value 10.
The size of the set of support for the Shirai contrast function.
See section 2.1

DO — Standard value 3.0
The initial value of D1. Any feature point within D1 units of its predicted location will be considered a GOOD POINT.
See section 3.2

D2 — Standard value 8.5
Any feature point greater than D2 units from its predicted location will be considered to be off the line.
See section 3.2

MA — Standard value 1.
A nervousness factor indicating how willing TRACK-LINE is to consider AMBIGUOUS POINT's to be GOOD POINT's after it has once failed to find any feature point.
See section 3.2

MN — Standard value 3.
Maximum value M1 can attain before causing TRACK-LINE to decide it has lost the line it was following.
See section 3.2

MT — Standard value 7.
Maximum value (+ M1 M2) can attain before causing TRACK-LINE to decide it has lost the line it was following.
See section 3.2

PHW — Standard value 20.
The half-width of the filter-values profile used to frame each suspected feature point.
See section 2.1

STEP — Standard value 10.0
The basic step size used for predicting where to look for the next feature point when following a line.
See section 3.2

WANDER-ANGLE
Standard value 30.0
Basic angle in degrees used to decide whether the direction of a line is straying too far from its initial estimate.
See Note 7.

WM Standard value 1.75
Scaling factor used to increase the value of D1 when successive AMBIGUOUS POINT's are found.
See section 3.2

Other special variables serve as actual variables for TRACK. In general, these should not be altered by the user but may be interrogated during debugging. They are summarized below:

CIRCLE-PTS the number of points in the circular profile last considered by FIND-SUSPECTS.

DOUBT-LIST the list of feature points currently considered by TRACK-LINE to be AMBIGUOUS POINT's.

D1 Any feature point less than D1 units from its predicted location is considered to be a GOOD POINT by TRACK-LINE. Any feature point between D1 and D2 units away from its predicted location is considered to be an AMBIGUOUS POINT by TRACK-LINE.

M the current enthusiasm factor used by TRACK-LINE to scale the basic step size STEP in predicting how far along the line to look for the next feature point.

M1 a parameter used in the algorithm of section 3.2 to record the current status of line following.

M2 a parameter used in the algorithm of section 3.2 to record the current status of line following.

PRFDGRAPH the current item number of the display item for the graph of the filter-values profile.

PRFDWIDTH the length of the filter-values profile (ie. (1+ (+ PHW PHW)))

RADIUS the radius of the last circular profile taken by FIND-SUSPECTS.

SCANITEM the current item number of the display item for the bands, circles, crosses, etc. associated with the current TRACK-LINE, VERIFY or FIND-SUSPECTS opration.

SCANWIDTH the length of the intensity values profile (ie. (1+ (* 2. (+ DELTA PHW))))

TYPE the type of the peak/feature-point/line last under consideration.

VIDGRAPH the current item number of the display item for the graph of the intensity values profile.

XC the x-coordinate of the vertex last considered in a FIND-SUSPECTS operation.

YC the y-coordinate of the vertex last considered in a FIND-SUSPECTS operation.

NOTES

1. For the purposes of this paper, a feature point will be said to be any point in the image array whose localized intensity profile "indicates" that the point might lie on a line.

2. Throughout the TRACK program package, intensity profiles are always taken beginning at the point rotated 90 degrees counter-clockwise from the direction of the line being tracked.

3. The values returned by our vidissector camera are a measure of the time taken for a fixed number of photons to image on the photocathode. Hence, the larger vidissector values represent darker points while the smaller vidissector values represent brighter points. (This is the usual source of confusion as to why a transition from high intensity to low intensity should be called a MAXIMUM.)

4. In the TRACK program, each entry in the intensity profile is actually the mean of the three intensity values taken along a band parallel to the direction of the line (ie. perpendicular to the direction of the profile) and centered at the indicated image point.

5. Throughout the TRACK program, line equations are parameterized in terms of θ, the angle the line makes with the positive x-axis, and ρ, the perpendicular distance of the line from the origin. Thus, the equation of a line is given by $\sin(\theta)X - \cos(\theta)Y + \rho = 0$. (See Horn{3} for more details.)

6. The TRACK program package NVSET's the vidissector to the highest resolution possible. For actual vidissecting, the x and y coordinates lie in the range 0 -> 16384. However, at all levels of interface with the user, TRACK makes use of the standard coordinate scale 0 -> 1024.

7. If the new equation of the line has an angle that is more than (MAX 5.0 (//$ WANDER-ANGLE n)) degrees different from the angle of the previous equation, the new equation will be rejected for prediction purposes. (WANDER-ANGLE is a special variable whose default value is WANDER-ANGLE = 30.0 and n is the number of points used in the calculation of the equation.)

8. WM is a special variable used simply as a scaling factor for determining the new value of D1 based upon the current values of DO and M2. (WM's default value is WM = 1.75)

9. It is important to realize that TRACK-LINE tracks line segments rather than lines. If the nature of a line changes along its path, TRACK-LINE, in general, will lose the line at the point of change. As an example, a particular obscuring edge may change from a MAXIMUM

(standard figure-background) to a MINIMUM (figure against a brighter background object) along its length.

10. We note, as an aside, that our line following algorithm is subject to the Mueller-Lyer illusion.

11. If a particular type is specified, only one of MAXLIST, MINLIST, HILIST or CRACKLIST could possibly receive any entries.

12. As in the case of the linear intensity profiles, the actual value used for each point in a circular intensity profile is calculated using the average of three radial points.

13. If T_i is MAXIMUM then T_{i+1} is MINIMUM (and vice-versa).

14. Since the directional sense of our circular profiles is opposite to that in the linear case, we reverse the sense of the peaks. (ie. a MAXIMUM in the circular filter-values profile corresponds to a MINIMUM feature point and vice-versa)

15. Before calling a feature point a HILITE or a CRACK, we want to be sure that the peaks of the corresponding MIN/MAX or MAX/MIN pair are sufficiently close together (where sufficiently close is interpretted to be that the peaks are within 3*DELTA vidissector units of each other, approximating the result for linear profiles).

REFERENCES

{1} Shirai, Yoshiaki, "A Heterarchical Program for Recognition of Polyhedra", AI Memo 263, AI Lab, MIT, June 1972.

{2} Herskovits, A., Binford, T., O., "On Boundary Detection", AI Memo 183, AI Lab, MIT, July 1970.

{3} Horn, B., K., P., "VISMEM: A Bag of Robotics Formulae", Vision Flash 34, AI Lab, MIT, December 1972.