# Hierarchical Generalized Hough Transforms and Line-Segment Based Generalized Hough Transforms*

Larry S. Davis
Computer Sciences Department
University of Texas
Austin, TX 78712

TN 80-6 November 1980

*This research was supported in part by funds derived from The National Science Foundation under grant ENG-7904037.

1. Introduction

The generalized Hough transform (GHT) is a fast point pattern matching procedure which has applications in computer vision - e.g., shape recognition. In its most general form, the GHT solves the following point pattern matching problem.

Given a set of object points $O \subseteq R^n$, a set of feature points, $P \subseteq R^m$ and a set of functions, $\mathcal{F}$, with $f \in \mathcal{F}$ being a mapping $f: R^n \rightarrow R^m$, find the $f \in \mathcal{F}$ such that $v(f) = |P - f(O)|$ is minimal. Here $|S|$ is the size of the set S, and if $O = \{O_1, \ldots, O_k\}$ then $F(O) = \{f(O_1), \ldots, f(O_k)\}$. By minimizing v(f) we guarantee that the maximal number of points in O are mapped onto points in P by f.

In all applications of the GHT considered to date, $n = m = 2$ and $\mathcal{F}$ contains translation, rotation and scaling maps from $R^2 \rightarrow R^2$. The set $\mathcal{F}$ is thus ordinarily determined by a set of parameters $T = \{t_1, \ldots, t_r\}$; for example, if $\mathcal{F}$ is the set of translation operators, then $T = \{t_1, t_2\}$ where $t_1$ is the x-translation and $t_2$ is the y-translation. We will let $f_{t_1 \ldots t_r}$ denote the function in $\mathcal{F}$ determined by the parameters $t_1, \ldots, t_r$.

In general, to apply the GHT, one must construct an r-dimensional array of accumulators, HT, (one dimension for each parameter in T) and impose a quantization and range restriction on that parameter space (which can be determined by an inspection of the points in P and O) so that HT is a discrete structure of finite size. The GHT procedure then assigns values to the array elements such that $HT(t_1,\ldots,t_r)$ is the count of the number of point pairs $(O_i,p_j)$, $O_i \in O$. $p_j \in P$ and $|f_{t_1 \ldots t_r}(O_i) - p_j| < \varepsilon$. The value of $\varepsilon$ is determined by the quantization of T.

The GHT procedure operates by considering all point pairs $(O_i,p_j)$, and computing all $t_1,\ldots,t_r$ such that $|f_{t_1 \ldots t_r}(O_i) - p_j| < \varepsilon$. This is ordinarily accomplished by considering all fixed quantized values of $t_1,\ldots,t_{r-2}$ and then computing the values of $t_{r-1}$ and $t_r$ (which are determined by the two points, $O_i,p_j$ in $R^2$) for which $|f_{t_1 \ldots t_r}(O_i) - p_j| = 0$; finally the nearest quantization levels for $t_{r-1}$ and $t_r(t'_{r-1},t'_r)$ are computed and $HT(t_1,\ldots,t_{r-2},t'_{r-1},t'_r)$ is incremented by unity. After all point pairs have been considered, the array HT must be searched for above-threshold peaks. The locations of those peaks indicate the possible instances of O in P.

Merlin and Farber[1] reported a GHT where $F$ was restricted to translations. Ballard [2] noticed that $F$ could also include rotations and scale changes, and contains a detailed discussion of the application of GHTs to computer vision. He points out that the manner in which both the feature point set, P, is ordinarily computed (by applying an edge detector to a digital picture) and the object point set is computed (by digitizing the boundary of a shape) can lead to drastic reductions in the number of elements in $O \times P$

which actually need to be compared when constructing HT, since in both cases orientation information can be associated with the elements of O and P.

Davis and Yam [3] suggested a similar GHT procedure to the one originally suggested by Ballard. However, they were concerned with the problems introduced by the size of the array HT, namely:

1) the storage requirements of HT, and
2) the computational requirements of searching HT for peaks.

To overcome these problems they suggested computing only an $r'$ dimensional projection of HT, $r' < r$. So, in their GHT procedure, all pairs $(O_i, p_j)$ are considered, and all r-tuples $(t_1,\ldots,t_r)$ for which $|f_{t_1 \ldots t_r}(O_i) - p_j| < \varepsilon$ are computed. However, rather than incrementing $HT(t_1,\ldots,t_r)$ in an r-dimensional array, one increments $HT(t_1,\ldots,t_{r'})$, $r' < r$, in an $r'$ dimensional array. They found that the GHT was still a robust shape matcher even in the projected parameter space, and discussed ways of recovering the parameter values lost due to projections by computing multiple GHT's in projected parameter spaces.

For example, if $T = (\Delta x, \Delta y, \theta)$ signifying transformations which translate and rotate, one can compute the GHT, $HT_1$, of P and O in an $\Delta x$, $\Delta y$ parameter space, and a second GHT, $HT_2$, of a translated version of O with P in the same $\Delta x$, $\Delta y$ parameter space. The relative location of the highest peak in $HT_1$ with respect to the highest peak in $HT_2$ can be used to recover the projected parameter, $\theta$.

In this paper we will consider two extensions to the GHT. The first is to hierarchical GHTs (HGHT), where the hierarchy is specified by a grammar. The advantage of the HGHT is added control over the distribution of O in P, i.e. rather than simply counting how many points in O match a point in P under $f \in F$, we can break O into pieces and additionally require that each piece match sufficiently. Of course, the pieces can be decomposed into pieces, etc. HGHTs are described in Section 2.

In Section 3 we consider a GHT procedure where P and O are not point patterns, but line segment patterns. This has obvious applications in computer vision where edges can be grouped into line segments (thus forming P) and planar shapes can have their boundaries well approximated by line segments (forming O). Thus, O x P may be much smaller for line segment patterns than point patterns, making the GHT that much more efficient.

## 2. Hierarchical GHTs

In general, there are two types of hierarchies which can be imposed on a shape representation: we will call them structural decomposition hierarchies (SDH) and resolution reduction hierarchies (RRH). The most prevalent SDH's are grammars. A shape grammar determines a hierarchical decomposition of a shape into finer and finer pieces, and specifies the geometrical and topological relationships between the pieces. A simple example of an RRH is a "pyramid." The pyramid is a set of regularly reduced resolution versions of the shape; each version represents a "snapshot" of the shape at some level of resolution.

The GHT can be extended to operate on the basis of both SDH and RRH shape representations. The principal advantage of using hierarchical representation is increased control over the shape recognition process. We will describe an SDH shape representation which will naturally lead to a hierarchical GHT algorithm.

If we are given a set of points, $S = \{(x_i,y_i)\}_{i=1}^{n}$, then a <u>Hough representation</u> based on S will be an ordered pair (S,c), where $c = (x_c,y_c)$ is an arbitrary point called the <u>center point</u>. It turns out that a convenient choice for c is the centroid of S, i.e. $x_c = 1/n \Sigma x_i$, $y_c = 1/n \Sigma y_i$. Now, suppose that we have the following:

1) a collection, $\mathcal{S}$ of sets of points, $S_j$, j=1,...,r, and

2) a <u>set synthesis grammar</u> (SSG), G, defined over the set $R = \{1,\ldots,r\}$. A set synthesis grammar G(R) is a 5 tuple, $< V_N, V_T, s, \mathcal{P}, M >$ where

1) $V_N$ is a set of nonterminal symbols,
2) $V_T$ is a set of terminal symbols, $V_N \cap V_T = \emptyset$, $|V_T| = r$
3) $s \in V_N$ is the start state
4) $\mathcal{P}$ is a set of productions of the form

$$Q \rightarrow x$$

with $Q \in V_N$, $x \in (V_N \cup V_T)^*$, Q does not occur in x.

5) M is a 1-1 map from $V_T$ onto $\{1,\ldots,r\}$

Let $x \in \mathcal{L}(G)$, where $\mathcal{L}(G)$ is the language of the SSG, G. Consider a syntax tree representation for a derivation of x in G. We will describe how to associate a Hough representation H(n), with each node, n, in the syntax tree. Tip nodes in the syntax tree are labeled with terminal symbols. If n is a tip node labeled with the terminal symbol v, then H(n) = $(S_{M(v)}, c_{M(v)})$.

Suppose that n is not a tip node, and that the sons of n are labeled with vocabulary symbols $v_1, v_2, \ldots, v_k$. If $H(v_i)$ is the Hough representation associated with node $v_i$, with center point $c_{v_i}$, then H(n) is defined as follows:

a) $c_n = \frac{1}{k} \sum_{i=1}^{k} c_{v_i}$

b) $H(n) = (\{c_{v_j}\}_{j=1}^{k}, c_n \}$

Thus, the point set associated with node n is the set of centers for the Hough representations of the immediate descendants of node n.

Figure 1 contains a simple example with $S_1 = \{(i,10)\}_{i=1}^{10}$, $S_2 = \{(10,i)\}_{i=1}^{10}$, $S_3 = \{(i,1)\}_{i=1}^{10}$ and $S_4 = \{(1,i)\}_{i=1}^{10}$; also $c_1 = (5.5,10)$, $c_2 = (10,5.5)$, $c_3 = (5.5,1)$ and $c_4 = (1,5.5)$.

Now, given $\mathcal{S}$, G (and, by construction, H(n), $n \in V_N \cup V_T$), a string $x \in L(G)$ and a set of points, P, we next describe how to detect instances of the pattern represented by x in P. First, a syntax tree, $T(x)$ for x in G must be constructed. The syntax tree will specify a partial order for the computation of Hough transforms at each node in the syntax tree. The GHT algorithm is as follows:

1) For each tip node, $v_i$, compute the Hough transform of $H(v_i)$ with respect to P. Let $F_i = \{f_{i,1}, f_{i,2}, \ldots, f_{i,r_i}\}$ be the locations of all above-threshold peaks of that Hough transform. Then $P_i = \{f_{i,1}(c_{v_i}), f_{i,2}(c_{v_i}), \ldots, f_{i,r_i}(c_{v_i})\} = \{P_{i,1}, \ldots, P_{i,r_i}\}$ is the set of potential <u>locations</u> of $v_i$ in P.

2) Let n be a non-tip node whose Hough transform has not yet been computed, but whose immediate descendants in $T(x)$, $v_1, \ldots, v_q$, have had their Hough transforms computed.

Let $P_i$, $i=1,\ldots,q$ be the sets of above threshold peaks for the Hough transforms of $v_i$, $i=1,\ldots,q$. We will write $H(n) = (C_n, c_n)$ with $C_n = \{c_{v_i}\}_{i=1}^{q}$. The Hough transform of n, HT, is computed as follows:

```
For each $c_{v_i} \in C_n$ do
    For each $P_{i,j} \in P_i$ do
        for all $f_{t_1 \ldots t_r} \in F$ such that
        $|f_{t_1,\ldots,t_r}(P_{i,j}) - c_{v_i}| < \varepsilon$ do
(*)     $HT(t_1,\ldots,t_r) := HT(t_1,\ldots,t_r) + 1$
    end
end.
```

Notice, that unlike the Hough transform for the tip nodes where every point in $C_n$ is compared to every point in P, for non-tip nodes a point $c_{v_i} \in C_n$ need only be compared with the possible locations of the ith son, i.e. the set of points $P_i$. Also note that at line (*) one could have incremented $HT(t_1,\ldots,t_r)$ by the value at $f_{i,j}$ in the Hough transform of $v_i$.

In [2], Ballard described a procedure for building the Hough representations of composite shapes, i.e. given two point patterns, $S_1$ and $S_2$, $H(S_1 \cup S_2)$ can be trivially constructed from $H(S_1)$, $H(S_2)$, $c_1$ and $c_2$ since one can simply compute a weighted average of $c_1$ and $c_2$ to obtain a center point for the set $S_1 \cup S_2$.

In contrast to Ballard's approach for building composite shapes, our hierarchical GHT provides for additional control over the detection of instances of a "composite shape" in a point set P, since using the grammatical approach described above, one must detect an instance of $S_1$ and an instance of $S_2$ before one can detect an instance of $S_1 \cup S_2$. If one simply detects $S_1 \cup S_2$ using a single composite Hough representation, then that degree of control is lost. We should also point out, however, that the price one must pay for that control is added computational cost. Computing the GHT of $H(S_1 \cup S_2)$ requires the same number of operations as computing the GHT's of $H(S_1)$ and $H(S_2)$. However, in the former case, only a single transform needs to be searched for above threshold peaks, while in the second case not only do two such transforms need to be searched, but the third, hierarchical transform (corresponding to $x \rightarrow S_1S_2$) needs to be computed and searched.

As an example of the application of hierarchical GHT's to image analysis we will consider an image registration problem. Figure 2 a-b contains two terrain model images, $f_1$ and $f_2$. An edge detector is applied to these images to produce the edge arrays $e_1$ and $e_2$ shown in Figures 2 c-d. Next, an "interest operator" is applied to the edge array in Figure 2c to detect 10, non-overlapping 11 x 11 interest areas. (See Figure 2e.)

The set of edge points in these interest areas constitute the point sets $S_i$, $i=1,\ldots,10$. The centroid, $c_i$, of each of these point sets is then used to construct the Hough representations, $H(S_i)$. The hierarchical GHT is then defined by the collection $\mathcal{S} = \{S_1,\ldots,S_{10}\}$ of point sets and the trivial SSG, $G_r = <\{S\}, \{S_1,\ldots,S_{10}\}, S, P, M>$ where:

1) $P = \{S \rightarrow S_1 \ldots S_{10}\}$ , and

2) $M: S_i \rightarrow i$

The center point for the Hough representation for S, then, is the centroid of the centroids of the $c_i$ (although the actual choice of the center point, recall, is arbitrary), and its point set is $\{c_1,\ldots,c_{10}\}$.

Once the Hough representations of the $S_i$ are computed, each is used to construct a Hough transform for the edge array in Figure 2d. The transform is computed with respect to the set of functions $F = \{f_{a_1,a_2}\}$ where $f_{a_1,a_2}(x,y) = x + a_1, y + a_2$. Thus, we only allow pattern translations. Five peaks are then chosen from the Hough transform of each interest point. Figure 2g shows the locations of the 5 best matches chosen for each interest area in Figure 2c. Let $\{t_{i,j}\}_{j=1}^{5}$ , be the locations of the 5 peaks for the i<u>th</u> interest area.

At this point, a variety of techniques can be chosen for assigning each interest area to a unique peak. However, computing another Hough transform is a particularly efficient technique for constructing this mapping. The transform we next compute will actually match the pattern of the $c_i$ in Figure 2e against the pattern of Hough transform peaks in Figure 2f. Thus, we compute the Hough transform HT(S), of H(S) with respect to the points $\{t_{i,j}\}_{i=1,}^{10} {}_{j=1}^{5}$. Note that in computing this transform as discussed previously, a point $t_{i,j}$ only needs to be compared with the point $c_i \in H(S)$ since prior information is available concerning to which point in S any point $t_{i,j}$ can correspond. If $c'_S$ is the location of the maximum of HT(S), then the vector $\overrightarrow{c_S - c'_S}$ indicates the displacement of frame $f_2$ from $f_1$. This displacement vector can be used to choose a best $t_{i,j}$ for each each $c_i$, or to predict that the interest area surrouding $c_i$ has either disappeared from the field of view, or has changed so substantially from $f_1$ to $f_2$ that it can no longer be seen in $f_2$. Figure 2g shows the final mapping of interest points to peaks. Yam and Davis [ 4 ] contain more details of the overall registration system, along with more examples.

## 3. GHTs for Line Segment Patterns

In this section we present extensions to the GHT procedure for point patterns which will allow for the matching of line segment patterns.

Let $L = \{L_1,\ldots,L_n\}$ be a set of line segments which represent the pattern of interest. $L$ corresponds to the object point set, $O$. Each $L_i$ is an ordered pair $(P_{s_i}, P_{t_i})$ where $P_{s_i} = (x_{s_i}, y_{s_i})$ is the <u>initial end point</u> of $L_i$, and $P_{t_i} = (x_{t_i}, y_{t_i})$ is the <u>terminal end point</u> of $L_i$. We will let $c_i = (\frac{x_{s_i} + x_{t_i}}{2}, \frac{y_{s_i} + y_{t_i}}{2})$ denote the midpoint of $L_i$, $\theta_i = \tan^{-1} (y_{s_i} - y_{t_i})/(x_{s_i} - x_{t_i})$ denote the orientation of $L_i$, and $\ell_i = \sqrt{(x_{s_i} - x_{t_i})^2 + (y_{s_i} - y_{t_i})^2}$ denote the length of $L_i$.

Let $L' = \{L'_i, L'_2, \ldots, L'_{n'}\}$ be a second set of line segments which corresponds to the feature point set, $P$. Here, each $L'_i$ is an ordered pair $(p'_{s_i}, p'_{t_i})$, and $c'_i$, $\theta'_i$ and $\ell'_i$ are the midpoint, orientation and length of $L'_i$, respectively.

Let $F$ be a set of functions $R^2 \rightarrow R^2$ which map line segments to line segments, and let the functions in $F$ be parameterized by the set $T = \{t_1,\ldots,t_r\}$.

Given two line segments, $L_i$ and $L_j$, we define their <u>difference</u>, $L_i - L_j$ as follows:

1) if $L_j$ is a subsegment of $L_i$, then $L_i - L_j = \ell_i - \ell_j$.

2) otherwise, $L_i - L_j = \ell_i$

Now, we can state the line segment matching problem as:

Given $L, L'$ and $F$, find $f \varepsilon F$ such that

$$v(f) = \sum_{L_i \varepsilon L} \sum_{L'_j \varepsilon L'} L_i - f(L'_j)$$

is minimal; i.e., f maps the maximal total length of line segments in L' onto subsegments of line segments in L.

The $f \varepsilon F$ which minimizes v can be computed using Hough transform procedures. For each pair of line segments $(L_i, L'_j)$, $L_i \varepsilon L$, $L'_j \varepsilon L'$, we determine all $f \varepsilon F$ such that $f(L'_j)$ is a subsegment of $L_i$ and then increment the appropriate location in the Hough transform by the length of $f(L'_j)$. Thus, when the computation of the Hough transform is completed, the location of maximal value will specify the function, f, with minimal v(f) since $\sum_{L_i \varepsilon L} \ell_i$ is independent of f.

Let us consider, as an example, the computation of the GHT for $F = \{f_{a,b}\}$, with $f_{a,b}(x,y) = (x+a, y+b)$. Now, given $L_i \varepsilon L$ and $L'_j \varepsilon L'$, we must determine all $f \varepsilon F$ such that $L_i - f(L'_j) \neq \ell_i$. Notice that if $\ell'_j > \ell_i$, or if $\theta_i \neq \theta'_j$, then no such f exists. If, on the other hand, $\theta_i = \theta'_j$ and $\ell'_j \leq \ell_i$ then there will be $(\ell_i - \ell'_j)/\gamma$ such functions f, where $\gamma$ is a constant which determines the quantization of the parameters a,b. In the algorithm which follows, $\mathcal{F}_{ij}$ is that subset of $\mathcal{F}$ satisfying $f \in \mathcal{F}_{ij}$ if and only if $L_i - f(L'_j) \neq \ell_i$. A simple digital difference analyzer (DDA) of the sort used in computer graphics to generate line segments can be used to compute $\mathcal{F}_{ij}$.

The GHT algorithm is then:

```
Procedure $HL_1(L,L')$
   for each $L_i \in L$ do
      for each $L'_j \in L'$ do
         if $\ell'_j \leq \ell_i$ and $\theta'_j = \theta_i$ then
            begin
               compute $\mathcal{F}_{ij}$
                  for each $f_{a,b} \in \mathcal{F}_{ij}$ do
                     $HT(a,b) := HT(a,b) + \ell'_j$
            end
      end
   end
end
```

As a simple example, consider Figure 3a-3c. Figure 3a contains a rectangle whose sides constitute the set L. The line segments in Figure 3b constitute the set L'. Figure 3c contains the transform of Figure 3b with respect to the rectangle in Figure 3a.

Consider H( 3,0) = 4. Which pairs of lines from L and L' contributed to H( 3,0)? First, when $L'_3$ is compared to $L_4$, then $f_{3,0}$ maps $L'_3$ into a subsegment of $L_4$ (in which their left end-points coincide); second, when $L'_4$ is compared to $L_2$, $f_{3,0}$ maps $L'_4$ into a subsegment of $L_2$ (in which their right end-points overlap). These are the only pairs of line segments, $(L'_j, L_1)$, $L'_j \varepsilon L'$, $L_1 \varepsilon L$, with $f_{3,0}(L'_j)$ a subsegment of $L_i$. Thus $H(3,0) = \ell'_3 + \ell'_4 = 3+1 = 4$

Algorithm $HL_1$ can be easily extended to perform rotation invariant matching. One simply deletes the test $\theta'_j = \theta_i$ at line 4 of the previous algorithm, and adds an inner loop surrounding lines 6-8 which first rotates $L_i$ through an angle, $\Psi$, before computing $F_{ij}$. More precisely, let $F$ again be the set of translations, let $L'_j(\Psi)$ be the result of rotating $L'_j$ through angle $\Psi$, and let $F_{ij}(\Psi)$ be the subset of $F$ which maps $L'_j(\Psi)$ onto a segment of $L_i$. Then the GHT algorithm for rotation and position invariant line segment pattern matching is:

```
1    Procedure $HL_2(L,L')$
2       for each $L_i \varepsilon L$ do
3          for each $L'_j \varepsilon L'$ do
4             if $\ell'_j \leq \ell_i$ then
5                for $\Psi = 0, 2\pi, d\Psi$ do
6                   begin
7                      compute $F_{ij}(\Psi)$
8                      for each $f_{a,b} \varepsilon F_{ij}(\Psi)$ do
9                         $HT(a,b) := HT(a,b) + \ell'_j$
10                     end
11                  end
12            end
13   end
```

## 4. Conclusions

This paper has attempted to make three points:

1) That the Hough transform shape matching algorithms recently proposed by Ballard [2] and Davis and Yam [3] are instances of a general point pattern matching algorithm;

2) That the point pattern matching algorithms can be usefully extended to match hierarchical point patterns, and

3) That the GHT can be further generalized to match patterns of geometric objects other than points, e.g. line segments. Clearly, the patterns can also include objects such as circular arcs, etc., and may even contain a mixture of geometric objects.

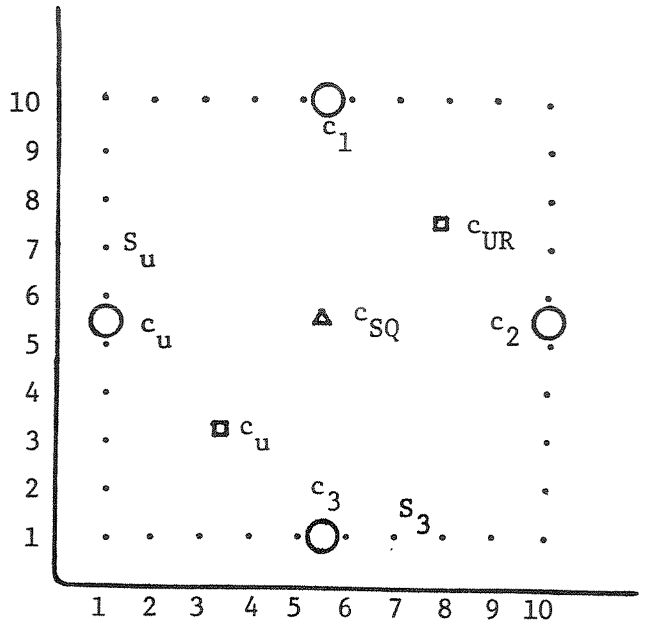
## References

1. P. Merlin and D. Farber, "A parallel mechanism for detecting curves in pictures," IEEET-Computers, 24, 94-95, 1975.

2. D. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," Pattern Recognition (to appear).

3. L. Davis and S. Yam, "A generalized Hough-like transformation for shape recognition," University of Texas Computer Sciences, TR-134, Feb., 1980.

4. S. Yam and L. Davis, "Image registration using generalized Hough transforms," University of Texas Laboratory for Image and Signal Analysis TN, in preparation.



$G = \langle V_N, V_T, S, P, M \rangle$

$V_N = \{LL, UR, SQ\}$

$V_T = \{1, 2, 3, 4\}$

$S = SQ$

$$P = \{ \begin{array}{l} SQ \rightarrow LL \ \ UR \\ LL \rightarrow 3 \ 4 \\ UR \rightarrow 1 \ 2 \ \} \end{array}$$

M :

| v | M(v) | H(v) | $c_v$ |
|---|---|---|---|
| 1 | 1 | $S_1$ | $c_1$ |
| 2 | 2 | $S_2$ | $c_2$ |
| 3 | 3 | $S_3$ | $c_3$ |
| 4 | 4 | $S_4$ | $c_4$ |
| LL | {3,4} | $\{c_3, c_4\}$ | $c_{LL}$ |
| UR | {1,2} | $\{c_1, c_2\}$ | $c_{UR}$ |
| SQ | {1,2,3,4} | $\{c_{LL}, c_{UR}\}$ | $c_{SQ}$ |

Figure 1

Figure 2a. Frame 1

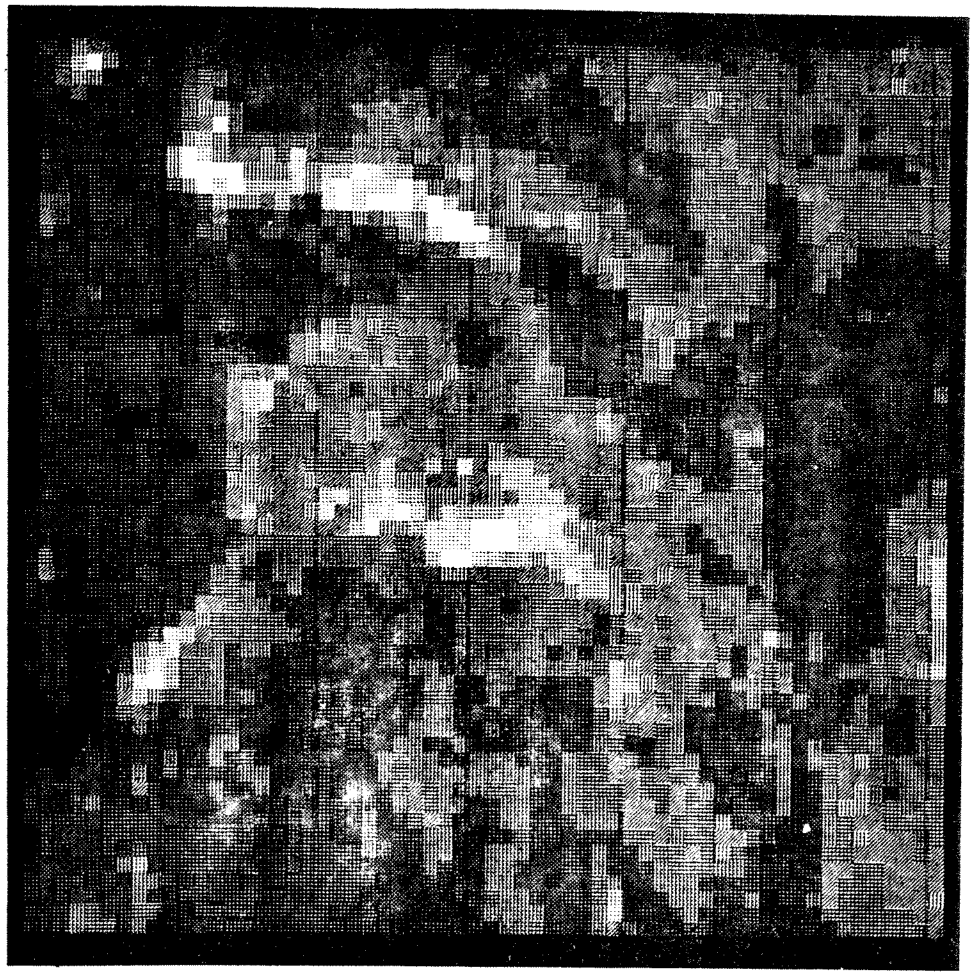


Figure 2b. Frame 2

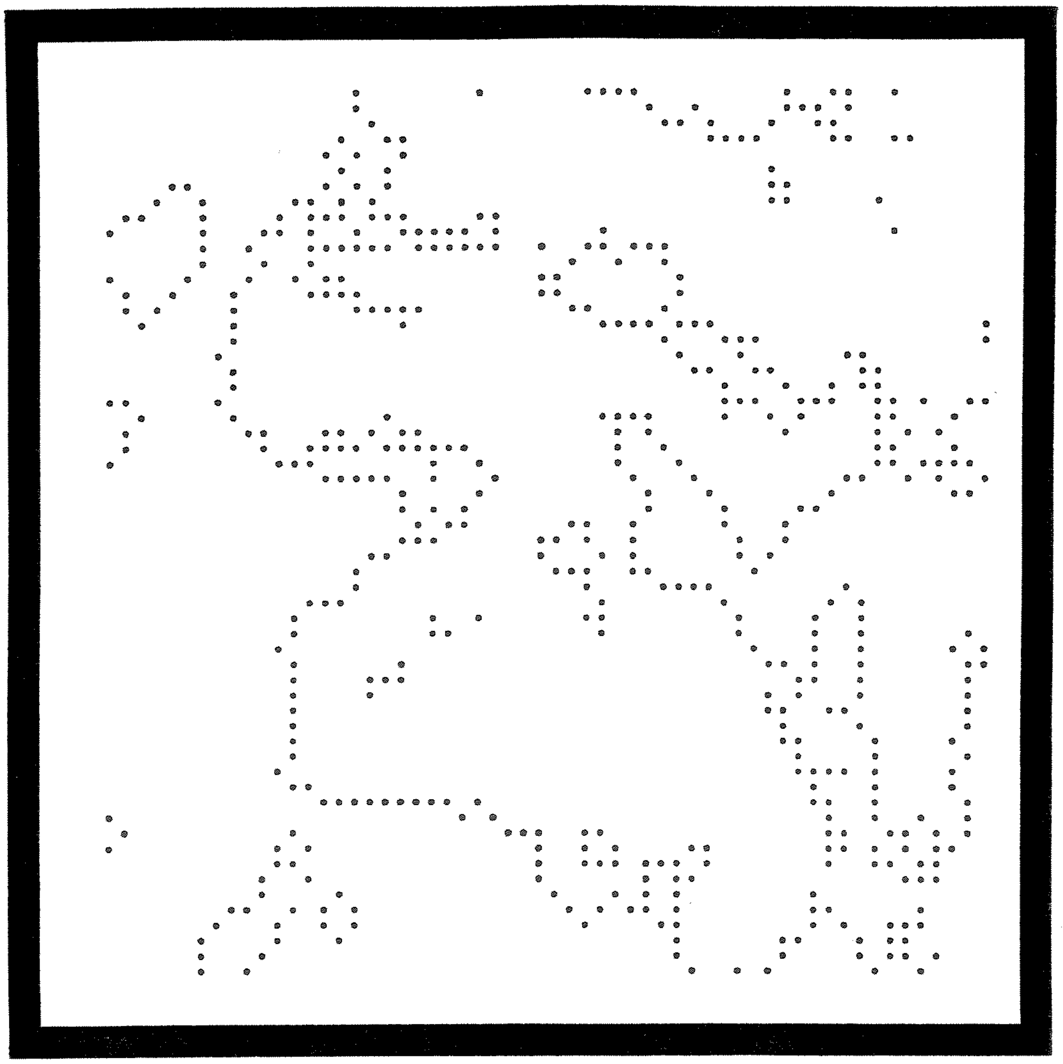


Figure 2c. Edge map for frame 1

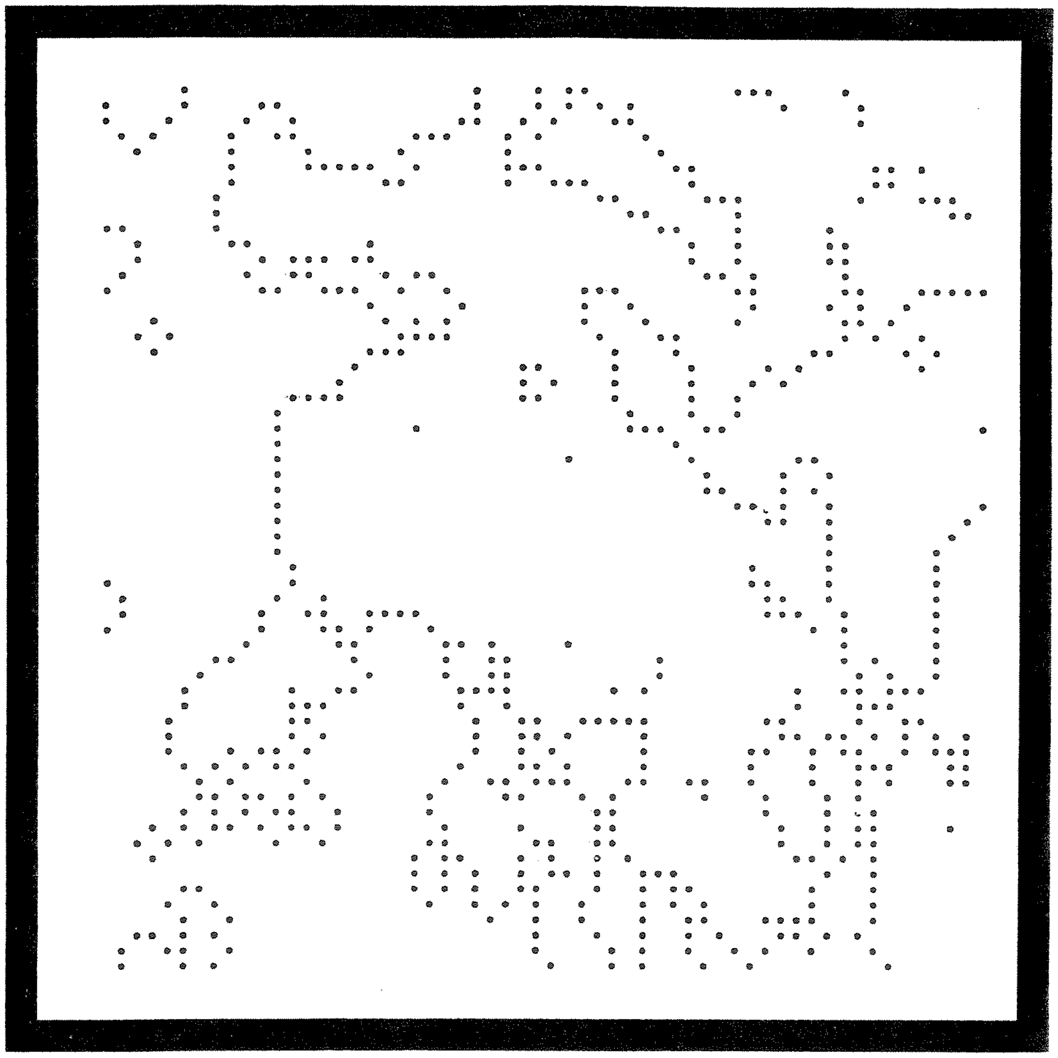


Figure 2d. Edge map for frame 2

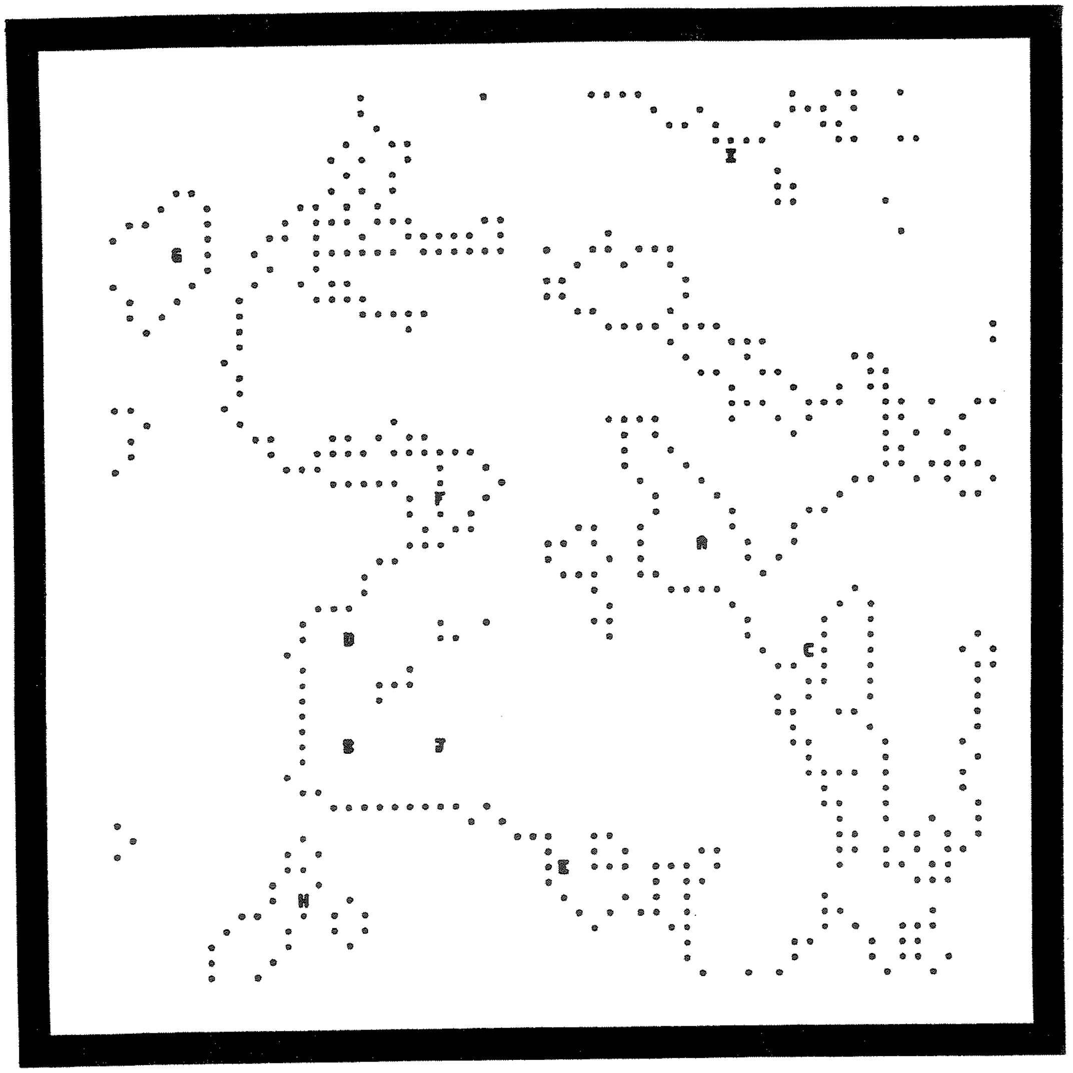


Figure 2e. Interest points for frame 1

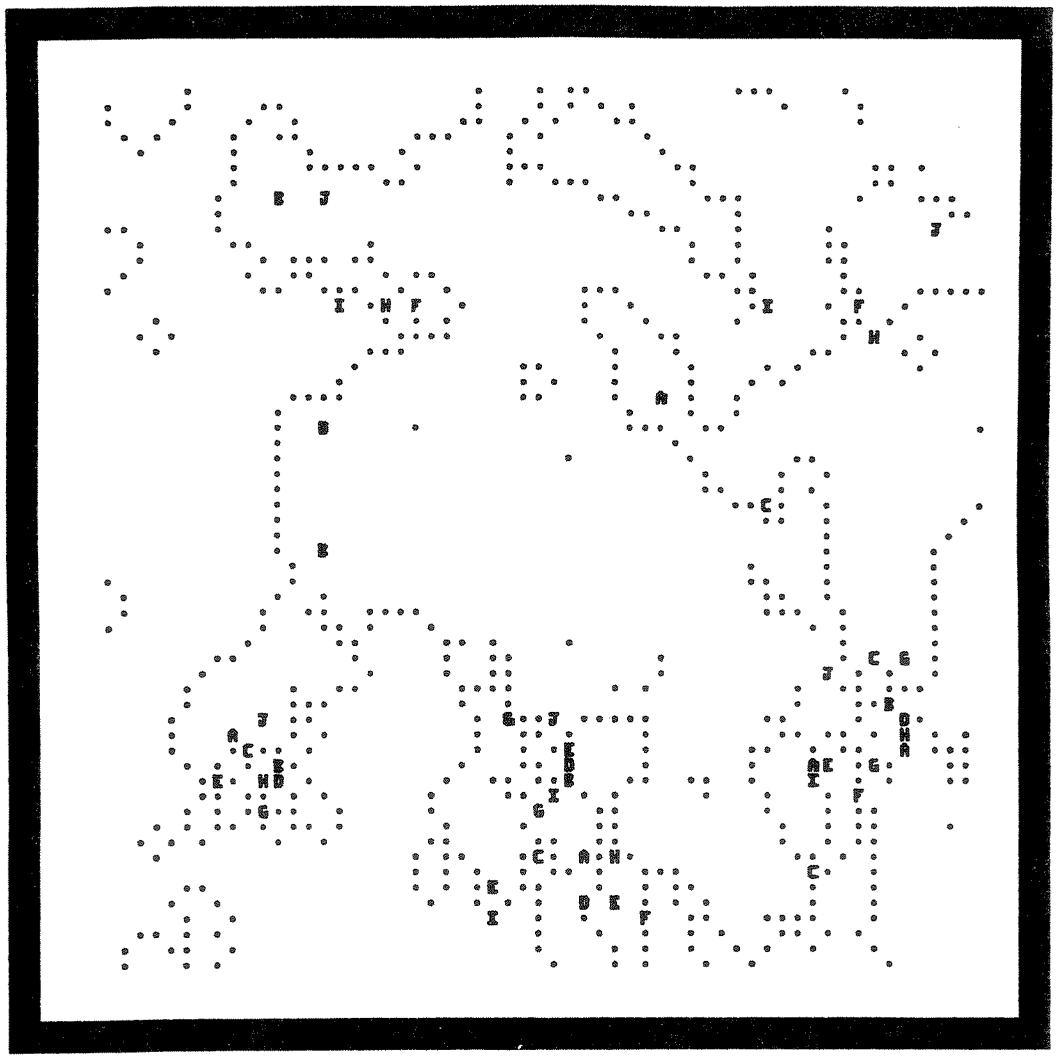


Figure 2f. Hough transform peaks in frame 2
for interest points in frame 1

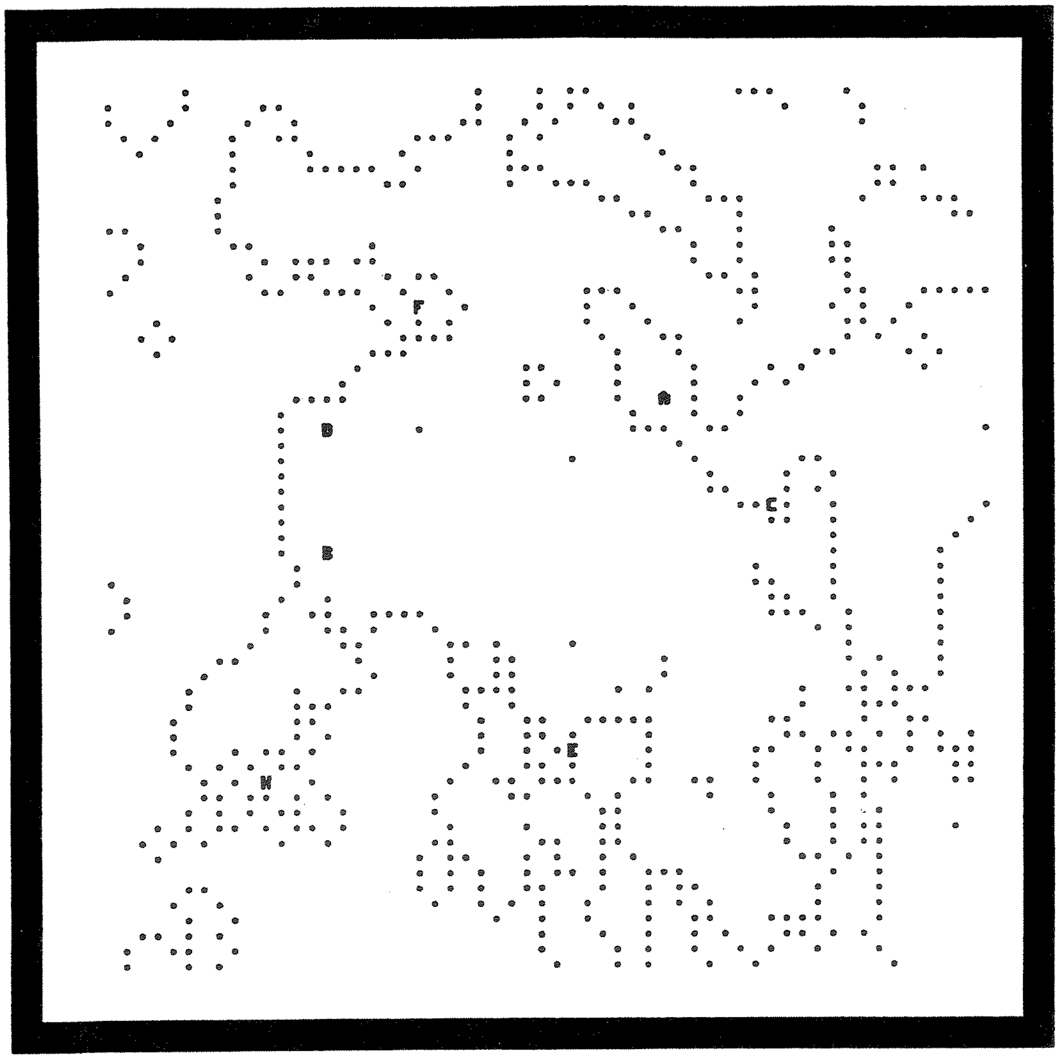
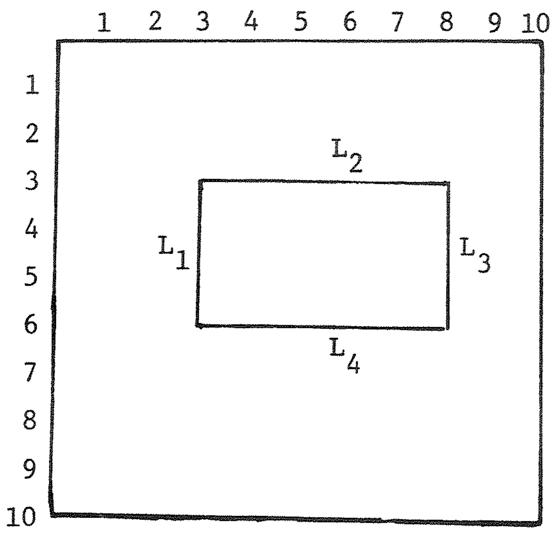
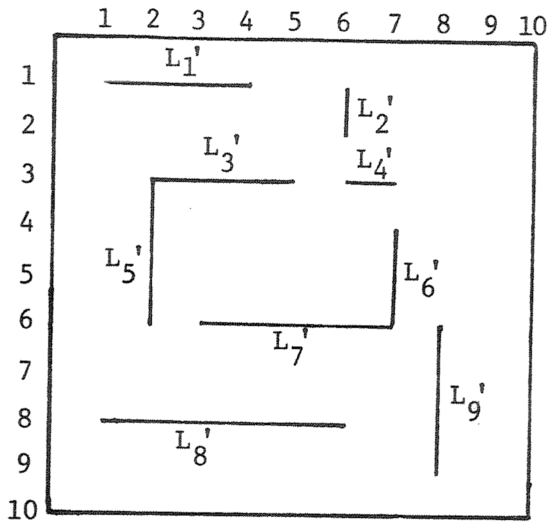


Figure 2g. Final registration of frame 2
to frame 1 interest points



a) Rectangle = L



b) L'

|  | -4 | -3 | -2 | -1 | 0 | -1 | -2 | -3 | -4 | -5 |
|---|---|---|---|---|---|---|---|---|---|---|
| -4 |  |  |  |  | 1 |  |  |  | 1 |  |
| -3 |  |  | 3 | 3 | 4 |  |  |  | 1 |  |
| -2 |  |  | 3 |  | 2 | 1 | 1 | 1 | 2 |  |
| -1 |  |  |  |  | 2 |  |  |  |  | 2 |
| 0 |  |  | 3 | 3 | 13 | 5 | 1 | 1 | 1 | 2 |
| -1 |  |  |  |  |  |  |  |  |  |  |
| -2 |  |  |  | 5 |  |  |  |  |  |  |
| -3 |  |  |  |  | 1 | 3 |  |  |  |  |
| -4 |  |  |  |  |  |  |  |  |  |  |
| -5 |  |  |  | 5 |  |  |  |  |  |  |

c) Hough transform-H

Figure 3. Example of Hough Transform