CORRESPONDENCE PROCESSES IN DYNAMIC SCENE ANALYSIS[1]

J.K. Aggarwal[2,3]
L.S. Davis[3]
and W.M. Martin[3]

TR-80-3 October 1980

[1]This research was supported in part by the Air Force Office of Scientific Research under grant AFOSR 77-3190 and by the National Science Foundation under Grant ENG-7904037.

[2]Department of Electrical Engineering
University of Texas
Austin, Texas 78712

[3]Department of Computer Sciences
University of Texas
Austin, Texas 78712

1. Introduction

Computational models for the analysis of time sequences of images of dynamic scenes are crucial for the solution of many image understanding problems. For example, in meteorology, the automatic prediction of frontal positions from satellite images of cloud cover requires that the movements of clouds be tracked from image to image [1]. In fact, meteorological applications imparted the initial impetus to research in motion analysis. The spectrum of applications has widened dramatically in the past several years to include biomedicine, tactical and strategic military applications and industrial automation. In addition to this variety of real world problems, models for motion analysis are of fundamental importance to our understanding of the human visual system. Mammalian visual systems not only contain "software" for motion analysis [2], but apparently also include "hardware" for the detection of moving objects in the visual field [3,4].

Perhaps the most prevalent problem in motion analysis is the tracking of objects from frame to frame. Tracking is a prerequisite for computing either the motion of the object or a description of how the object is changing. A general approach to tracking is to establish correspondences between points, or sets of points, in successive frames and then to group those sets into objects based on similarity of motion. Such grouping operations are often based on the assumption that the objects are <u>rigid</u> or that they are articulated (i.e. jointed) but composed of rigid parts. Such assumptions impose structural constraints on the relative positions of object points, which in turn impose constraints on the two-dimensional pattern of positions projected by those points onto the image plane. For example, Ullman [5] and Roach and Aggarwal [6] use the rigidity assumption to compute the three-dimensional structure of moving objects from multiple views of the object in motion, while Webb [7] and Rashid [8] discuss how jointed objects may be analyzed. It is important to note that in a great number of applications the objects in motion can be treated as two-dimensional so that grouping operations and accompanying computational models of motion can be specified solely on the basis of changes in image coordinates.

This paper will focus attention on processes for establishing the correspondence between sets of points in successive frames. It should be pointed out that these correspondence processes have applications to other problems in image understanding besides motion analysis - e.g., stereopsis, change detection, etc. We will not consider the subsequent grouping procedures which establish structure and motion from such correspondences. Discussions of those problems can be found in [5-6].

There are a number of factors which contribute towards making the correspondence problem quite difficult; the presence or absence of such factors determines the procedures which can be applied to solve any specific correspondence problem. First, the types of transformations that objects can be subject to from frame to frame must be considered. Can the objects change their orientation in the field of view, or their size? Can their shape change, and if so, is any prior information available which constrains such changes? (This is especially important for tracking clouds, which change shape, sometimes dramatically, from frame to frame.)

Objects may be moving against changing backgrounds, and this tends to complicate the correspondence processes. It is much simpler to track an object which is moving against a clear, blue sky than it is to track an object which is moving along the ground from one type of textured region to another. Furthermore, if it is possible for the object to move behind other objects, so that it is only partly visible at times (or even completely invisible for some time) then the correspondence processes must be able to establish their matches given only partial information.

We will consider two general approaches towards establishing a correspondence between image parts in successive frames. The first is based on constructing "iconic" or picture-like models of a segment of one frame. Such iconic models are sometimes referred to as <u>templates</u> in the picture processing literature. Early psychological theories of visual perception attempted to account for human pattern recognition based on iconic memory models. However, such theories fail to account for recognition of patterns that are highly abstract, or generalized (e.g., the recognition of caricatures, cartoons, etc.). Similarly, computational theories of pattern recognition based on iconic pattern representations are not applicable in all situations. Nevertheless, the computational efficiency of algorithms for matching iconic models against images justifies their serious consideration before more general modelling techniques are adopted.

The second approach employs structural models for segments of the first frame, and then computes homologous representations for segments of the second frame against which it matches those pieces. Such procedures can, in general, tolerate grosser changes in size, shape, etc., than can procedures based on iconic models. They are, however, computationally more complex.

2. Tracking Using Iconic Models

In tracking using iconic models, one must first "lock-on", or detect, a subset of the first frame which one suspects contains a moving object, construct an iconic representation of that subset, and then match that iconic representation against the second frame. Suppose that for the frame acquired at time t_1, $F_1 = f(x,y,t_1)$, $x_1 \leq x \leq x_2$, $y_1 \leq y \leq y_2$ is a subimage that contains a moving object. For F_1 we define $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$. Then there are a variety of iconic representations which can be constructed based on F_1. These include:

1) Using F_1 directly;

2) Segmenting F_1 into a binary image (i.e., an image composed of 0's and 1's), where the 1's indicate object points and the 0's non-object points; and

3) Applying an edge detection operation to F_1 resulting in a binary image where the edges, or boundaries, of the objects are labeled 1, and all other points are labeled 0.

The advantage of using F_1 directly is that it requires the minimal amount of computation to construct the iconic representation. However, the exact gray levels in F_1 depend not only on the properties of the moving object (e.g. its reflectance and shape), but also on the properties of the background against which it is moving. If this background can change dramatically, relative to the object, from frame to frame (e.g., if the background is textured), then it might prove difficult to match F_1 based upon this direct representation.

On the other hand, segmenting F_1 into either an object/background or edge/no-edge representation makes the salient shape characteristics of the moving object explicit in the iconic representation. One must be able, however, to compute such segmentations reliably. In the remainder of this section we will denote by F_1' the iconic representation of F_1.

Now, suppose that $F_2 = f(x,y,t_2)$, $0 \leq x \leq n$, $0 \leq y \leq m$, is the frame acquired at time t_2. In order to match F_1 against a piece of F_2(i.e. a subset of F_2 having the same shape and size as F_1) one must first compute an iconic representation of that subset which is of the same form as the representation chosen for F_1. We will let F_2' refer to that iconic representation of F_2.

Next, one must adopt some measure of match between F_1' and F_2'. The measures may be either <u>similarity</u> measures (high values indicate match) or <u>difference</u> measures (low values indicate match). A variety of such measures have been considered including the following:

1) Normalized cross correlation (similarity measure)

$$C(x,y) = \frac{P(x,y)}{Q(x,y)}, \text{ where} \tag{1}$$

$$P(x,y) = \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_1(x_1+i,y_1+j)\ F_2(x+i,y+j) \text{ and}$$

$$Q(x,y) = \sqrt{\sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_1(x_1+i,y_1+j)^2 \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_2(x+i,y+j)^2};$$

2) Sum of absolute differences (difference measure)

$$A(x,y) = \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} |F_1(x_1+i,y_1+j) - F_2(x+i,y+j)|; \text{ and} \tag{2}$$

3) Sum of squared differences (difference measure)

$$S(x,y) = \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} [F_1(x_1+i,y_1+j) - F_2(x+i,y+j)]^2. \tag{3}$$

The normalized cross correlation is closely related to the sum of squared differences since

$$\begin{aligned} S(x,y) &= \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} [F_1(x_1+i,y_1+j) - F_2(x+i,y+j)]^2 \\ &= \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_1(x_1+i,y_1+j)^2 \\ &\quad -2 \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_1(x_1+i,y_1+j)\, F_2(x+i,y+j) \\ &\quad + \sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_2(x+i,y+j)^2 \end{aligned}$$

and if $\Sigma\,\Sigma\, F_1^2$ and $\Sigma\,\Sigma\, F_2^2$ were fixed, then $S(x,y)$ would be minimized when

$$\sum_{i=o}^{\Delta x} \sum_{j=o}^{\Delta y} F_1(x_1+i,y_1+j) \quad F_2(x+i,y+j)$$

is maximized. This latter quantity is the <u>unnormalized cross correlation</u>. One must normalize it as above because the unnormalized cross correlation will be high in areas of F_2 having high average intensity, whether or not they match F_1. For $C(x,y)$, it is easily shown that $0 \leq C(x,y) \leq 1$, and that $C(x,y)=1$ if and only if for some constant c

$F_1(x_1+i,y_1+j) = c\, F_2(x+i,y+j)$ $0 \leq i \leq \Delta x$, and $0 \leq j \leq \Delta y$ [9].

From the point of view of computational expediency, the difference measures (2) and (3), are preferable to the similarity measure (1) because their <u>cumulative</u> nature (see below) allows them to be incorporated into fast matching algorithms.

The straightforward space domain algorithm for computing any of the preceding match measures requires $\Delta x \Delta y$ operations per pixel. If Δx and Δy are large, then this can take a significant amount of computing time on a general-purpose computer. We will discuss two approaches toward reducing this time:

1) The use of special-purpose computer architecture; and,

2) the development of faster algorithms for general purpose computers.

There are at least three distinct types of architectures for image processing which can profitably be distinguished; each is accompanied by its own specific set of advantages, disadvantages, and theoretical and practical problems.

1) "Focal plane architectures" which are actually integrated into video sensors behind the focal plane and which are capable of processing data at high-quality television data rates (7.5 MHz).

2) Cellular arrays of simple, bit serial processing elements (PE's). Cellular arrays are a special class of single-instruction stream multiple-data stream (SIMD) machines having fixed-interconnections.

3) General multiple-instruction stream multiple-data stream (MIMD) machines with many general-purpose processors, memories and a flexible interconnection network. Such machines can also be operated as SIMD machines.

As one moves from architectures of type 1 through type 3, there is a significant decrease in speed. Focal plane architectures can compute a relatively complex computation (e.g., a 5x5 convolution) at the rate of 100 ns/pixel, while a cellular array such as CLIP 4 [10] and MIMD machines (such as ZMOB [11] would operate at significantly lower data rates (see Davis [12] for more details).

Balancing this decrease in processing efficiency is an increase in processing generality. Focal plane architecture is functionally quite rigid; it cannot, e.g., be used to apply iterative algorithms to an image unless the number of iterations is known a priori. Even then, it requires duplication of circuitry, (e.g., the median of median operation computed by TI VLSI architecture [13]). The cellular arrays are more general, since their PE's are ordinarily capable of computing any Boolean function over a single bit plane of a point and a simple function of its four or eight neighbors. However, for non-logical operations, the PE's are very difficult to program due to their "low-level" instruction set. MIMD machines composed of many microprocessors are still more general, since not only are the microprocessors' machine instructions ordinarily quite powerful, but compilers are available for translating high-level languages (such as PASCAL or FORTRAN) into the machine language of the microprocessors. However, difficult problems in scheduling and sharing need to be solved before MIMD machines become generally available.

For the purposes of object tracking using iconic matching techniques, focal plane architectures would be most preferable because they can support such computations at close to real-time data rates. As one example of a "focal plane architecture" for convolutions, consider the approach suggested by Texas Instruments [14] based on VLSI technologies.

In general, the correlation of a sequence $X = \{X_i\}_{i=0}^{m}$ with a sequence of weights $W = \{W_i\}_{i=0}^{n}$ is defined by

$$C(i) = \sum_{j=0}^{n} W_j X_{i+j} \quad . \tag{4}$$

This is essentially the one-dimensional form of the unnormalized cross correlation discussed above. It is possible to extend the design discussed below to normalize C(i) by $\sum_{j=o}^{n} X_{i+j}$, but the principle point of the design is the efficient computation of C(i). Now if we express X_n as

$$X_n = \sum_{b=0}^{r} X_{n,b}*2^b \quad , \quad X_{n,b} \in \{0,1\} \tag{5}$$

then by substituting (4) into (5) and reordering terms we can obtain

$$C(i) = \sum_{b=0}^{r} \left(\sum_{j=0}^{n} W_j X_{i+j,b} \right) 2^b \quad .$$

Thus, C(i) can be computed using a total of about rn shifts and adds. However, time can be saved by <u>prestoring</u> all values of

$$\sum_{j=0}^{n} W_j X_{i+j,b}$$

in a 2^n by $B_w + \log_2(n)$ bit read-only memory (ROM) where B_w is the number of bits required to store the maximum W_j. Now, the computation of C(i) takes r+1 table look-ups in the memory, and r+1 shifts and adds. This technique is called the ROMaccumulator (RAC) technique.

An advantage of TI's VLSI design is that the dynamic range of the convolution weights can be increased with only a small increase in ROM. On the other hand, the VLSI approach is impractical for large convolutions. Even a small, 10x10 convolution would require a ROM which is $2^{100}x(B_w+\log_2 n)$ bits, which is clearly impossible. If one adopted the blocking schemes suggested in [14] (i.e. essentially break the large memory into several smaller memories and then combine the results with additional circuitry), then the architecture might become too slow. Note that one is also faced with the formidable problem of loading the partial product memory (which for image tracking could obviously not be constructed with ROM). This requires both <u>computing</u> all the partial products, and then <u>storing</u> them into memory. Such problems need to be faced before such architectures could be applied to tracking problems. See [15] for an alternative architecture based on charge-coupled devices.

An alternative to using special purpose architecture is to design fast algorithms for computing the location of maximum match. Although it is possible to use frequency domain techniques, we will restrict our attention to space domain techniques because they generalize to wider classes of match functions.

Barnea and Silverman [16] introduced a class of fast algorithms for image registration which avoided the comparison of every point in F_1' with every point in F_2'. In the following, we present a generalization of some of the ideas presented in [16] which involves representing the matching problem using state-space representation techniques, and then searching for the best match using an ordered search algorithm. For notational convenience, we will develop the algorithm using only one-dimensional pictures.

Let $q(i)$, $o \leq i \leq m$, be a sequence of numbers which represents a one-dimensional image and let $p(i)$, $o \leq i \leq n, n < m$ represent a one-dimensional object whose position in q we want to detect. We say that the sequence $p = \{p_i\}_{i=o}^{n}$ is an <u>initial sequence</u> of a second sequence $p' = \{p_i\}_{i=o}^{n'}$ if

a) $n \leq n'$ and

b) $p_i = p'_i \quad o \leq i \leq n$.

Let M be any <u>cumulative</u> <u>mismatch</u> function for matching a sequence $p = \{p_i\}_{i=o}^{n}$ against a sequence $q = \{q_i\}_{i=o}^{m}$. M is <u>cumulative</u> iff when $p = \{p_i\}_{i=o}^{n}$ is an initial subsequence of $p' = \{p'_i\}_{i=o}^{n'}$ $M(p,q(j)) \leq M(p',q(j))$. Here, $M(p,q(j))$ will be a difference measure that represents the dissimilarity of the sequence p to the subsequence $q_j, \ldots, q_{j+n-1}$.

As an example, consider the mismatch function, A, defined as

$$A(p,q(j)) = \sum_{i=o}^{n} |p_i - q_{i+j}| .$$

If p is an initial subsequence of p', then we can write p' as $p_1 p_2 \ldots p_n \; p'_{n+1} \ldots p'_{n'}$. Clearly,

$$A(p',q(j)) = \sum_{i=o}^{n'} |p'_i - q_{i+j}|$$

$$= \sum_{i=o}^{n} |p_i - q_{i+j}|$$

$$+ \sum_{i=n+1}^{n'} |p'_i - q_{i+j}|$$

$$\geq \sum_{i=o}^{n} |p_i - q_{i+j}| = A,(p,q(j))$$

so that A, is a cumulative mismatch measure.

The state-space, then, is defined as follows: A <u>state</u> is a triplet $\langle t,j,M(p^t,q(j))\rangle$ where

1) t indicates how long an initial subsequence of p has been compared against q starting at position j,

2) j is a position in q,

3) p^t is the initial subsequence of length t of p, and

4) $M(p^t,q(j))$ is the dissimilarity of p^t to q(j).

A start state is of the form $\langle r,j,M(p^r,q(j))\rangle$, where $1 \leq r \leq n$ and for each $r, 1 \leq j \leq (m-r)$, while a goal state is one for which r=n. Notice that the start state represents a situation where an initial subsequence of p has been compared to q at position j, while a goal state represents the situation when <u>all</u> of p has been compared to q starting at position j. If $S = \langle t,j,M(p^t,q(j))\rangle$ is a state, then the k-successor of S (denoted $O_k(S)$) is S' where

$$S' = \langle t+k,j,M(p^{t+k},q(j))\rangle \quad ;$$

i.e., S' is obtained by comparing k more points from p against the subsequence of q beginning at j. If $S_r = O_k(S_{r-1})$, $S_{r-1} = O_k(S_{r-2}),\ldots,S_2 = O_k(S_1)$, then $S_1,\ldots,S_r$ is a <u>path</u> from S_1 to S_r. The <u>cost</u> of the path is the value of the dissimilarity measure in state S_r (which is also the maximum of the dissimilarity measures for the set of states $S_1,\ldots,S_r$); we will also refer to this cost as the cost of S_r. The objective, then, is to find a minimum cost path between a start state and a goal state. This can be accomplished by an ordered search algorithm [17]. The cumulative nature of the mismatch algorithm assures the admissability of the algorithm - i.e., it is guaranteed to find a minimum cost path.

The ordered search algorithm is defined as follows:

1) Put all start states, $<r,j,M(p^r,q(j))>, o\leq j\leq m-n$, into a set called OPEN.

2) Choose the state from OPEN with minimal dissimilarity measure and delete it from OPEN. Let $S=<t,j,M(p^t,q(j))>$ be this state.

3) If S is a goal state, then the best match of p to q occurs at position j, and the algorithm halts. Otherwise, continue.

4) Compute $O_k(S)$ and add this new state to OPEN. Go to Step 2.

A slight modification of the above algorithm, employed by Barnea and Silverman [16], can lead to dramatic savings in computation time; however, the algorithm would no longer be admissable. The modification involves not putting into OPEN any state $<t,j,M(p^t,q(j))>$ with $M(p^t,q(j))>T(t)$, where T is a threshold function. Barnea and Silverman [16] discuss methods for computing a reasonable threshold function from the sequences p and q. We will not adopt this modification in the example below.

As an example of the application of the algorithm, consider Figure 1. Figure 1a contains an image, q, and an object,

p. The iterations of the ordered search algorithm are described in Figure 1b, using k=1, r=2 and dissimilarity measure Δ, the sum of absolute differences. The example proceeds as follows: With r=2, the initial subsequence of p of length 2 (i.e., 4 7) is matched against the subsequence of length 2 in q starting at each of the positions 0,1,...,5. For example, at position 3, the mismatch is $|4-3| + |7-2| = 6$. For each partial match a state is entered into the set OPEN. The minimal cost state is S=<2,1,1>.

With k=1, $O_k(S_1)$ is obtained by adding the mismatch of p_2 and q_3 to the cost of S_1. This mismatch is $|2-3| = 1$, so that $O_k(S_1)$ is the state S_2=<3,1,2>, indicating that the initial subsequence of length 3 of p (i.e., 4 7 2) has been matched to q_1 q_2 q_3. This state, S_2, is added to OPEN.

S_2 is also the minimal cost state at iteration 2, with $O_k(S_2)$ being the state S_3=<4,1,4>. Notice that even though S_3 is a final state, the algorithm does not terminate. For a final state to be chosen by the algorithm it must be known to be minimal over the current OPEN set, thus S_3 must first be placed in OPEN and its dissimilarity measure compared to the measures of all states. At iteration 3 this comparison yields the minimal cost state S_4=<2,4,3>. $O_k(S_4)$ is the state <3,4,6>, but at iteration 4 state S_5=<2,5,4> is chosen arbitrarily from the minimal cost subset of OPEN, {<2,5,4>, <2,0,4>, <4,1,4>}. We next choose <2,5,4> whose successor, <3,5,6>, is then placed in OPEN. Then, at iteration 5, <2,0,4> is taken from OPEN, and its successor, <3,0,9>, is placed on OPEN. Finally, at iteration 6, goal state <4,1,4> is chosen from OPEN and the algorithm halts.

Although the ordered search algorithm will decrease the number of comparisons of points in p with points in q, there are two sources of overhead which might render the strategy more costly than the direct approach:

1) The algorithm must maintain a sorted list of states representing partial matches of p to q; and

2) If not enough primary storage is available to simultaneously maintain all of q, then the algorithm may need to page pieces of q in and out when the subsequence of q associated with the newly chosen state of Step 2 is found not to be in the currently available storage. This I/O overhead can severely degrade the performance of the algorithm.

The computational cost of matching p against q can also be reduced by employing a <u>subtemplate</u>-<u>template</u> matching strategy [18]. Here, one matches a piece, p' of p against q, and then matches the remainder of p only at those points in q where the match of p' is sufficiently good (e.g., higher than some threshold, t). If p' has n' points, n'<n, then the total amount of computation performed by the subtemplate-template matching algorithm is

$$W(t) = mn' + mnP,$$

where P is the probability that the match of p' to q(j) has a value greater than t for randomly chosen values of j. Note that unlike the ordered search strategy, the subtemplate-template matching strategy is <u>not</u> guaranteed to find the best match since there is a non-zero probability that the match of p' to q at the position which maximizes the match of p to q will be below the threshold, t. Note that this false dismissal rate can be kept arbitrarily close to the false dismissal rate of matching p to q in two ways:

1) lowering the threshold, t, for matching p' against q or

2) increasing the size of the subtemplate, p'.

Lowering the threshold, of course, will increase $W(t)$ since for $t'<t$, more points in q will match p'. If t is made too low, then it is possible for $W(t)$ to be greater than mn, which is clearly undesirable. Similarly, increasing the size of p' may also increase $W(t)$. Vanderbrug and Rosenfeld [18] discuss choices of n' and t which minimize $W(t)$ while keeping the overall error rate below threshold.

A related strategy to subtemplate-template matching is <u>coarse</u>-<u>fine</u> template matching [19]. Here, one first matches an averaged and sampled version, p', of p against a similarly averaged and sampled version, q', of q. Positions in q' which are good matches to p' are then used to guide the application of p to q. Again, there are trade-offs between reliability and two factors - the size of p' relative to p and the threshold used in matching p' to q'.

The coarse-fine matching strategy can be further generalized to matching in a <u>pyramid</u> image representation [20,21]. A pyramid is a stack of regularly reduced resolution versions of an image. Tanimoto and Pavlidis [22] for example, describe an edge detection procedure which operates in a pyramid.

The applicability of these correlation-based matching procedures is limited by a number of factors. The two most important of these for object tracking are the inability to deal with objects whose orientations in the image plane change from frame to frame and the inability to match given only partial information. The structural techniques discussed in Section 3 are designed to overcome these problems.

3. Structural Matching Techniques

In this section we shall discuss methods which establish the correspondence between points of interest at consecutive time instances using structural models or domain constraints to guide the process. The points of interest are assumed to be derived from the images by low level operators which can detect specified components and determine the locations and descriptive feature values of those components. Each such component, together with its features will be referred to as a token. For example, a simple 3x3 edge operator with local non-maxima suppression could be used to form a token representing an edge which is considered to be centered at a given pixel with a specific orientation and contrast. The function of the matching process is thus to construct a mapping from the set of tokens of one image to the set of tokens of a second image. Clearly, the methods suitable for establishing this mapping depend on the particular attributes retained with the tokens.

However, inter-token constraints imposed either by structural models or the scene domain are also important. Object models can be derived from two primary sources. General descriptive models of the objects or object types expected to occur in the scene can be provided to the analysis system before processing is initiated. In this case the tokens in each image are matched against the descriptive features contained in the models. For a given token in one image the corresponding token in the preceding image is identified as that token which matched the same model feature as the given token. Models can also be derived from the images as they are processed. In this case general information about object formation is used to group the tokens in an image into structures which are a first estimate of the object models and provide constraints useful for establishing the correspondence to the tokens in another image. Such scene domain constraints can also be applied to individual tokens, usually in the form of limits imposed on the area of search for matching tokens.

An early system which employed motion measurements for scene segmentation (Potter [23]) formed tokens referred to as "cross-shaped templates." The attributes of these tokens were the distances (horizontal and vertical) from a given pixel to the nearest gray level discontinuity. To match a given token of this sort from one image a heuristic search of the second image was performed, starting at the image location of the original token. The search expanded outward from that starting position and continued until either a similarity measure over the token attributes exceeded a threshold, i.e., a match was found, or a pre-set search limit was reached. The displacement between the locations of matched tokens constituted the motion measurement for the tokens of the first image. The segmentation of that image was then performed using the constraint that tokens with the same motion measurements were part of the same object.

Two major problems arose for the system. First, the attributes associated with the tokens limited the allowable object motions to be simple translations in the image plane. Second, the system attempted to form a token for every pixel in the image plane, thereby incurring extensive computation in the search phase. By considering the distance from each object boundary point, i.e., the gray level discontinuities used in [23], to some central position, a model of the object to be tracked can be used in conjunction with a matching technique to overcome these problems. The matching technique is a generalization of the Hough transform (Duda and Hart [24]) to arbitrary shapes as encoded in boundary list representations (Ballard [25], Davis and Yam [26]) and will be discussed in the following. We will first describe position invariant matching, and then describe generalizations to orientation and scale invariant matching.

Let $B = \{(X_i, Y_i)\}_{i=0}^{n}$ be a list of boundary points for the shape to be tracked. B might be the set of edge locations in an image window detected by an "interest operator" at time t_1. Let $p = (X,Y)$ be <u>any</u> point (in practice, a central point such as the centroid of B will be computationally convenient to use as p). Then the Hough-representation of B using p, H(B,p), is the sequence of vectors $\{d_i\}_{i=0}^{n}$, where $dx_i = X-X_i$ and $dy_i = Y-Y_i$.

Now, suppose we are given an image, f, which contains an instance of the shape whose boundary is described by B. Here, f would be the image acquired at time t_2. A second array, h, which is an array of accumulators that is registered with f, will be used to compute the transform of f with respect to H(B,p). After the transform is computed, points in h with high values will correspond to hypothetical locations of p in f. Of course, once the location of p is known, the instance of B in f can be recovered using H(B,p). The array h will be larger than the array of f, since if the shape is only partly contained in f, the point p might lie outside of f.

The transform, h, is computed by first applying an edge detector to f to produce an edge map, e, of f. Each edge, e_i, in e is a potential element of the set B. Although contrast and orientation information may limit the subset of B to which any e_i may correspond, there is, in general, no way to determine the element of B to which any e_i corresponds without considering the positions of all the other e_i. Therefore, each edge element, e_i, is compared to each vector in H(B,p) to compute a possible location for p, and that location is incremented in the transform, h. That is, h is computed by the following simple algorithm originally reported in [25].

```
Algorithm MATCH 1:
For each $e_i = (X_i, Y_i)$ in e do
   For each $d_j = (dx_j, dy_j)$ in H(P,p) do
      $h(X_i+dx_j, Y_i+dy_j) :=$
         $h(X_i+dx_j, Y_i+dy_j) + 1$.
```

Notice that the result of applying this algorithm is exactly the same as correlating a binary image representation of B with the binary edge map, e (this was originally pointed out by Sklansky [27]). The correlation, however, is based on considering all points in h as potential locations for p, and then for each location counting the number of appropriately positioned (according to H(B,p)) edges in e. The advantage of the transform algorithm is computational efficiency. If h is an rxs array then to compute h using a standard correlation algorithm requires O(rxsxn) operations - i.e., for each of rxs potential locations for p, we must check the n locations of possible edge points determined by H(B,p). Algorithm MATCH 1, on the other hand, requires O(|e|xn) operations, where |e| is the number of edges detected in f. Since, in practice, edges account for no more than 5%-10% of any image, algorithm MATCH 1 will result in speed-ups of 10 to 20 over conventional correlation procedures. As an example of this process, consider the two aerial photographs contained in Figures 2 and 3. The objects in these images appear to move toward the top of the image. Figure 4 shows the edge points found in Figure 2, along with points of interest (marked by letters) derived by grouping edge points into sets, one set per point of interest. A Hough-representation is formed for the set of edge points associated with each point of interest using the location indicated by the letter as the central point. There are ten points of interest, i.e. tokens, in Figure 4. For each of these tokens a Hough transformation relative to the edge points found in the second image is formed. Figure 5 shows the edge points for the second image in addition to the positions of the five highest peaks in each transform. Intertoken constraints were then employed to determine the "best" matches. Note that two tokens, G and I, moved off the image and another token, J, disappeared due to structural changes in the image. In these three cases the "best" match was determined to be no match.

In the preceding, we assumed that the orientation of B in f was known. Suppose, on the contrary, that it is not known (this can occur, e.g., while tracking a vehicle, from above, which is moving along an unpredictable path). In this case, when we hypothesize that a particular e_i corresponds to some d_j, the strongest conclusion we can draw is that if e_i were indeed d_j, then p must lie somewhere on the circle of radius $R_j = \sqrt{dx_j^2 + dy_j^2}$ centered at e_i. The following algorithm accomplishes rotation invariant matching.

```
Algorithm MATCH 2:
For each $e_i = (X_i, Y_i)$ in e do
   For each $d_j$ in H(B,p) do
      $R_j = \sqrt{dx_j^2 + dy_j^2}$
   For $\theta$ =0, $2\pi$, by $d\theta$ do begin
      $h_x = R_j * \cos \theta + X_i$;
      $h_y = R_j * \sin \theta + Y_i$;
      $h(h_x, h_y) = h(h_x, h_y) + 1$;
end.
```

Unlike algorithm MATCH 1 where the results were identical to what could have been obtained by correlating the binary image e with a binary image representation of B, the results of applying algorithm MATCH 2 are not identical to what would be obtained by individually correlating $m = 2\pi/d\theta$ rotated versions of B with e, and then choosing the maximum match amongst the m correlation planes. Instead, algorithm MATCH 2 adds the m correlation planes together to obtain a single plane (h). The position in this plane having maximum value is then interpreted as the location of B.

Notice that if prior information is available concerning the orientation of the object in the frame, then this information can be easily taken advantage of by the algorithm. One simply modifies the bounds on the inner FOR loop so that only circular arcs in h, rather than entire circles, are incremented. For example, in tracking vehicles moving along roads, one can ordinarily assume that between the successive frames the vehicle will not make a turn sharper than $\pi/2$, since roads do not bend that quickly.

Although algorithm MATCH 2 can detect an arbitrarily oriented version of a shape, it does not compute the orientation of the shape. This could be done by maintaining m separate correlation planes and applying algorithm MATCH 1 to m rotated versions of H(B,p). In practice, however, this approach has unacceptable storage and time requirements.

Instead it is possible to construct a second transform of B, but with respect to a different point, p'. If (i,j) is the point in the transform of H(B,p) having maximal value, and if (i',j') is the point in the transform of H(B,p') having maximal value (notice that these values must, in principle, be identical), then the direction from (i,j) to (i',j') gives the direction from p to p' in f. Points p and p' should be chosen to be sufficiently far apart so that small errors in the locations of the maxima in the transforms h of H(B,p) and h' of H(B,p') do not lead to large errors in the computed orientation of B.

Notice that the algorithms can also be modified in a straightforward way to deal with a limited range of scale information. Suppose, e.g., it is known that the object in the image is S times the size of the model, with $S \in [S_1, S_2]$ (note $S_1 < S_2$ and $0 < S_1$). Then in algorithm MATCH 1 rather than just incrementing a single point at distance $d = \sqrt{d_x^2 + d_y^2}$ from an edge point, one marks all points in direction $\tan^{-1}(d_y/d_x)$ and with distances $d' \in [S_1 d, S_2 d]$. For rotation invariant matching, rather than incrementing a circle (or circular arcs if constraints on the orientation are available) one increments a ring of inner radius $S_1 d$ and outer radius $S_2 d$ (or the intersection of the ring with wedges). Again, different correlation planes can be maintained for different values of the scale, but this increases the storage and computational requirements of the matching algorithms. Note that this idea was employed by Davis [28] to detect circles of various sizes using Hough transform techniques.

Boundary descriptions of objects were also employed by the system discussed in Martin and Aggarwal [29]. This system extracted simple closed curves representing figures with curvilinear boundaries from each image in a time ordered sequence. In this case the input was restricted so that the figures independently moved in planes parallel to the image plane. The figures were planar with opaque homogeneous shading. This meant that when the figures moved so as to occlude one another the boundaries merged into apparently single figures. The main task of the system was thus to derive descriptions of the actual figure boundaries which were constituents of the apparent figures in the images.

The fact that a given figure in the image might be composed of boundary points from several actual figures precluded the matching of the entire boundary between two images. Instead the boundaries were broken into sets of tokens with each token representing a boundary section which approximated a straight line or a circular arc. The token attributes were the length and curvature of the represented arc. The matching process began by finding pairs of highly similar tokens, referred to as "seeds." The remainder of the matching process made use of the ordering of the tokens on the figure boundaries to constrain the segment "growing" algorithm that was applied to the already matched "seeds."

This process was able to detect extended boundary segments having the same shape in consecutive images. Since the actual figures were assumed to be rigid, a pair of matched segments could be interpreted as being two views of a portion of an actual object boundary. Thus the boundary shapes were used to form the correspondence which in turn provided motion measurements for each matched segment. The final grouping into actual objects was based on the constraint that segments exhibiting similar motions were sections of a single rigid object.

This latter constraint is quite important and is the basis of the object interpretation in most current motion analysis systems. However, for the correspondence processes in the systems discussed so far in this section, the matching has been based on token attributes not related to motion. In the remainder of this section we will illustrate how the motion or the expected motion of the scene components can be used in forming the correspondence.

Endlich, et al. [30] did not incorporate an explicit movement expectation but did assume that most of the tokens within arbitrarily chosen subimages exhibited similar velocities. Under this assumption their system formed a correspondence which specified a consistent velocity for the largest number of tokens in a subimage. The tokens were referred to as "brightness centers" and had an intensity attribute as well as an image location. The procedure used by the system was to iteratively refine the estimate of the representative velocity and to use the estimated velocity to constrain the possible matches for each token. The process was iterated until each token had no more than one possible match.

The velocities determined for each subimage were then merged together to yield a velocity map which represented the cloud motions in the satellite images processed by this system. For more general image sequences, however, the arbitrary partitioning of the image into subimages could be a severe problem because the presence of two or more independently moving scene components in a given subimage would invalidate the assumption of a representative velocity for that subimage. In most cases it will be impossible to know, a priori, how to partition the image so that each subimage contains only one scene component. One might, however, make use of the localized motion consistency constraint in a network of competing hypotheses, much like the consistent labelling procedures of Rosenfeld, et al [31].

Barnard and Thompson [32] accomplished this by locating the prominent (i.e., most likely matchable) features in each of a pair of images with an "interest operator". Associated with the operator was a similarity measure which was used both to initialize the "probabilities" for the hypothesized matches at the tokens and to update those "probabilities" at each iteration of the refining process. The network upon which this refining process operated was created from the tokens in the first of the pair of images by establishing a node for each token and connections between all nodes whose tokens were within a pre-set distance of each other. At each node a "label" set was formed containing an element for each possible match of the associated token. The labels were ordered pairs of the disparity, in the x and y directions of the image plane, between the location of the token of the node and the location of each token within a given radius of that position in the second image of the pair.

In addition to the possible match labels, there was included a special label specifying that no match could be found for the given token. The inclusion of this special label is indicative of an important concept for motion analysis systems: the tracking procedures must be robust enough to continue functioning properly when some of the feaures of interest, i.e., the tokens currently being tracked, are no longer detectable in the image. A particular feature may not appear in a given image of the sequence for several reasons: the feature might become occluded by scene components of the foreground; the image characteristics might for some reason, e.g., illumination changes, not remain within the tolerances of the interest operator; or the feature may move beyond the view of the imaging device.

The motion consistency constraint of Barnard and Thompson [32] was propagated throughout the network by increasing the probability assigned to a given label at a specified node by an amount proportional to the sum of the probabilities of the similar labels at connected nodes. This label refining process could be iterated until the network stabilized or until every node had a clearly defined "most likely" label. However, in practice it was iterated ten times, leaving a few ambiguous labellings.

A network of a different sort was proposed by Ullman [5]. Here, a node was associated with each token in the pair of consecutive frames. For each token in one frame the set of possible matches in the other frame was determined and connections were established between the node for the given token and the nodes for the tokens in the matching set. This network was used to calculate the correspondence which minimized a mapping cost function. The calculation was to be performed by simple processors, one attached to every node and connection in the network. Each node processor communicated with the processors attached to the connections incident on that node, while each connection processor communicated with the processors at the two nodes which terminated the connection. Thus the processors could be partitioned into three disjoint sets: one set associated with the tokens of the first frame; another set related to the tokens in the second frame; and the final set representing possible matches between two tokens, one from each frame.

All processors computed only simple functions of values stored at the neighboring processors and used those functions to update their own values. This updating procedure was iterated until the values in the network stabilized, at which time the connection processors had values of either 1 or 0, only. The resulting correspondence was then specified by the set of connection processors that have a value of 1, i.e., a given token of the first frame was mapped to a token of the second frame if the nodes for those tokens were connected in the network and the processor attached to that connection had a value of 1. Thus, the specified correspondence was the mapping which yielded the minimal cost.

The cost function proposed was directly related to the probability distribution of the velocity of the tokens as measured in the images. Minimizing this cost function was shown by Ullman [5] to be optimal under the assumption that the movement of each token was independent of the movements of the other tokens. It was also argued that one-to-one mappings should be preferred and then shown that a simple modification to the cost function would effect this preference.

As the time between frames increased and the velocities of the tokens decreased, the mapping also tended to minimize the total distance moved by all tokens in the scene. This was the case in which the nearest neighbor match of the tokens tended to be a one-to-one mapping. At higher velocities nearest neighbor matching would result in numerous "splits" and "fusions", i.e., one-to-many and many-to-one mappings, while the "optimal" cost minimizing mapping would retain its preference for one-to-one mappings. In this way, although the nearest neighbor match might not be the desired correspondence, it could be used as the initial estimate for the minimizing process which derives the "optimal" mapping. Thus the network would be constructed by connecting the node of a given token in one frame to the N nodes associated with the N nearest tokens from the other frame. It should be noted that there is no mechanism for adding connections to the network once the minimization process has begun, so the initial set of possible matches would have to contain the "correct" one.

There were two major problems with the overall scheme proposed in Ullman [5] for forming the correspondence between frames. The first was similar to the problem of subimage selection of Endlich, et al. [30], in that the cost function used a single distribution for the velocities. This was justified by the independence of movement assumption. Clearly, the validity of this assumption would be in doubt if several tokens were established for each object in the scene. In that case the motions of the tokens associated with a particular object would be interdependent and directly related to the overall movement of the object. The second problem was the requirement that the correspondence be specified by a "cover", i.e., every token in both frames was matched. Note that for the examples studied in Ullman [5] this was not a problem because the images were of dot patterns in which occlusion was rare. However, for general scenes the features that give rise to tokens will frequently disappear and appear necessitating the no-match possibility, as discussed earlier in this section. A solution to this problem might be to introduce into the original network two special nodes for which the connection would mean "no match." Initially all the nodes for a given frame would be connected to one of the special nodes and all the nodes for the other frame would be connected to the remaining special node. The difficulty here would be determining the cost associated with the no match connection as it relates to the velocity distribution and the one-to-one mapping preference.

The expected velocity of a token was also used in the correspondence forming process of the system described in Rashid [8]. Again, the input was a sequence of images of dot patterns with a token created for each dot. In this case, however, each token retained its own expected velocity parameter. The expectations were used to determine the predicted locations for the tokens from one frame. Those computed locations were then matched against the tokens in the next frame. The desired correspondence was the one-to-one mapping from the set of predicted locations to the set of tokens for the new frame which minimized the sum of all the distances between the predicted locations and their matched tokens.

The minimization was not computed by a network of simple processors. Instead, a Voronoi construction (Shamos [33]) was used to provide an efficient implementation. For a set of N locations a Voronoi construction tessillates the plane into N polygons (some possibly infinite in size). Each polygon is associated with exactly one of the locations in the set and bounds the area containing all the points for which the associated location is the closest element of the set. Thus to determine the closest location to a point, one need only ask which polygon of the Voronoi construction contains that point. In Rashid [8] a Voronoi construction was performed for the set of predicted locations from a given frame, then each token of the new frame was matched to the closest predicted location by finding which polygon included the token. If more than one token was within a single polygon then the token nearest to the associated predicted location was chosen as the match for that location. The matched location was deleted from the set and a new Voronoi construction was computed. The efficiency lies in the fact that given a Voronoi construction for N points, computing a new construction using N-1 of those points is linear in the number of points.

The problems that occur with this scheme are twofold and concern the initialization and updating of the velocity prediction functions. First, since the prediction is used to form the correspondence, the mapping from the first frame of the sequence to the second frame must depend on expectations supplied to the system or on some default expectations. The validity of these initial expectations is important because an incorrect yet consistent mapping between the first pair of frames will generate erroneous predictions for mapping the third frame, and so on.

The second problem might occur when a token changes its velocity. This was partially accounted for in Rashid [8] by making the expected velocity be the average of the two velocities measured from the immediately previous frames. This works well if the velocity "varies smoothly," as assumed in [8]. However, abrupt changes in velocity would invalidate the prediction and leave the system with the problem of determining an initial prediction again. These are crucial points for any predictive scheme: the prediction mechanism must have an initialization phase, a normal updating phase, and an error detection and correction phase.

A similar sort of predictive analysis was used in the top level of the hierarchical matching system described in Roach and Aggarwal [34]. The input domain for this system was that of images of polyhedra moving in three-dimensional space. The images were processed to yield the edges of the visible planar surfaces, with line and vertex descriptions derived from these extracted edges. These descriptions were then segnented into preliminary object interpretations based on general domain constraints. The domain, however, did allow ambiguities which generated multiple interpretations of given line-vertex groups. These interpretations were maintained by the system until conclusive evidence was obtained to decide which was correct.

The correspondence between consecutive images was established by a hierarchical system which invoked the lower level processes only as the upper levels failed. The top level process calculated a centroid for each object interpretation and using information from preceding images determined a predicted location for every centroid. These predictions were then matched by a nearest neighbor rule to the centroids found in the succeeding image. As long as the predictions remained valid this level was sufficient, otherwise the system invoked the second level matching process.

At the second level, coarse descriptions of relative object positions, e.g. object A is to the left and below object B, were used to match object interpretations. The coarseness of the feature ensured that the description would remain constant for fairly long intervals of time. However, upon failure, the third and lowest level matching process was activated. This process matched object interpretations based on the relative positions of the polygonal faces in each interpretation. In this manner several different levels of processes used information from various object descriptions and relationships to establish the correspondence between images.

This section has described several methods for operating upon points of interest extracted or abstracted from the gray level information in the images to form an inter-image correspondence. These methods employed scene domain constraints, structural information from object models, constancy features of the moving objects, and predictive analysis based on movement expectations. The complexity of the movements analyzable by the methods is greater than those of Section 2, while, the analysis requires the images to contain distinct and discrete features from which the tokens can be created.

4. Summary

We have discussed procedures for solving the correspondence problem based on both iconic and structural representations of the image parts to be matched from frame to frame. The iconic representations lead to fast matching algorithms, but are not general enough to be applied to all correspondence problems. Matching algorithms based on structural representations, while ordinarily more demanding computationally than iconic-based algorithms, can tolerate a wider variety of pattern transformations (e.g., rotations, scale changes, etc.)

The correspondence problem, is, of course only one in a sequence of problems that must be solved in dynamic scene analysis. The principal other problems, which this survey did not address, include the detection of motion, the grouping of moving parts into objects, and the recognition and tracking of those objects from frame to frame.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
q::=	6	5	7	3	2	6	5	4	8

	p_0	p_1	p_2	p_3
p::=	4	7	2	4

a) An image, q, and an object, p.

Iteration	OPEN	S
1	{<2,0,4>, <2,1,1>, <2,2,6>, <2,3,6>, <2,4,3>, <2,5,4>}	<2,1,1>
2	{<2,0,4>, <2,2,6>, <2,3,6>, <2,4,3>, <2,5,4>, <3,1,2>}	<3,1,2>
3	{<2,0,4>, <2,2,6>, <2,3,6>, <2,4,3>, <2,5,4>, <4,1,4>}	<2,4,3>
4	{<2,0,4>, <2,2,6>, <2,3,6>, <2,5,4>, <4,1,4>, <3,4,6>}	<2,5,4>
5	{<2,0,4>, <2,2,6>, <2,3,6>, <4,1,4>, <3,4,6>, <3,5,6>}	<2,0,4>
6	{<2,2,6>, <2,3,6>, <4,1,4>, <3,4,6>, <3,5,6>, <3,0,9>}	<4,1,4>
7	HALT	

b) Trace of the algorithm applied to the data in 1a.

Figure 1. Example of the ordered search algorithm for one-dimensional images.

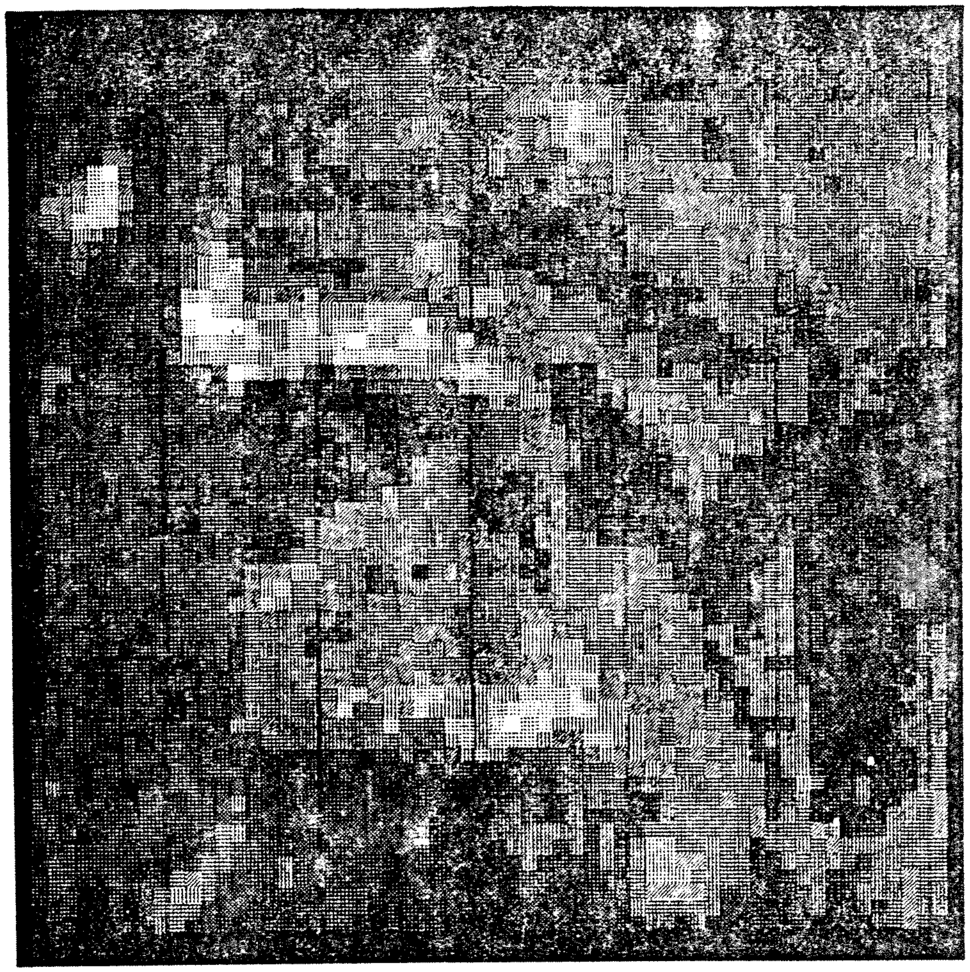

Figure 2. First image of pair to be matched.

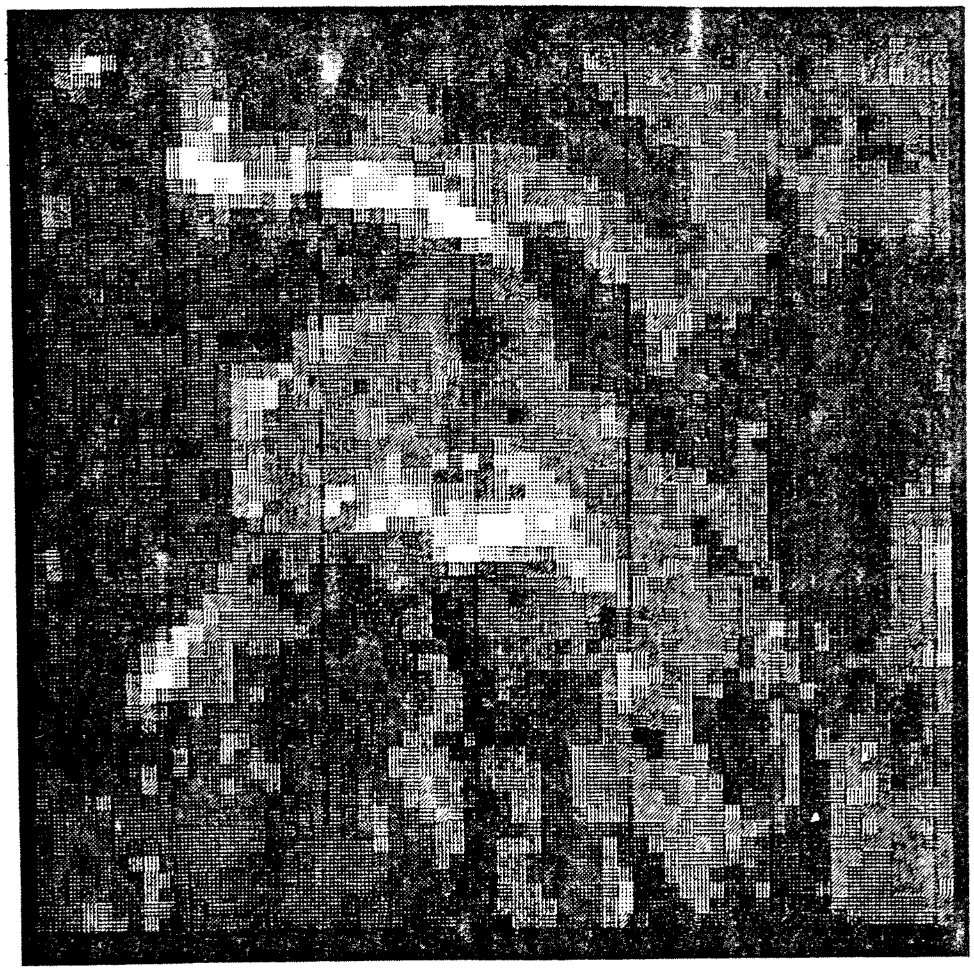

Figure 3. Second image of pair to be matched.

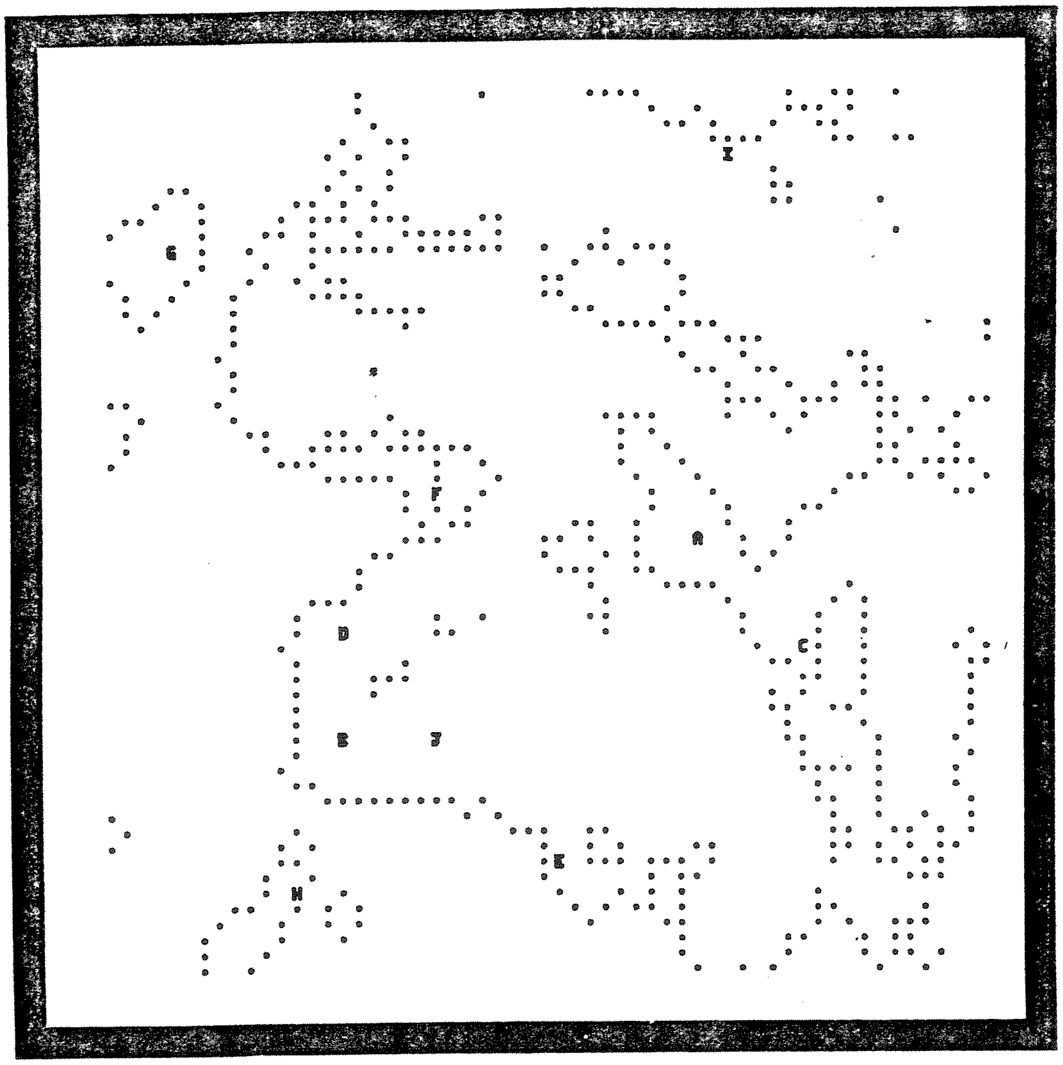

Figure 4. Edge points and points of interest derived from the image in Figure 2.

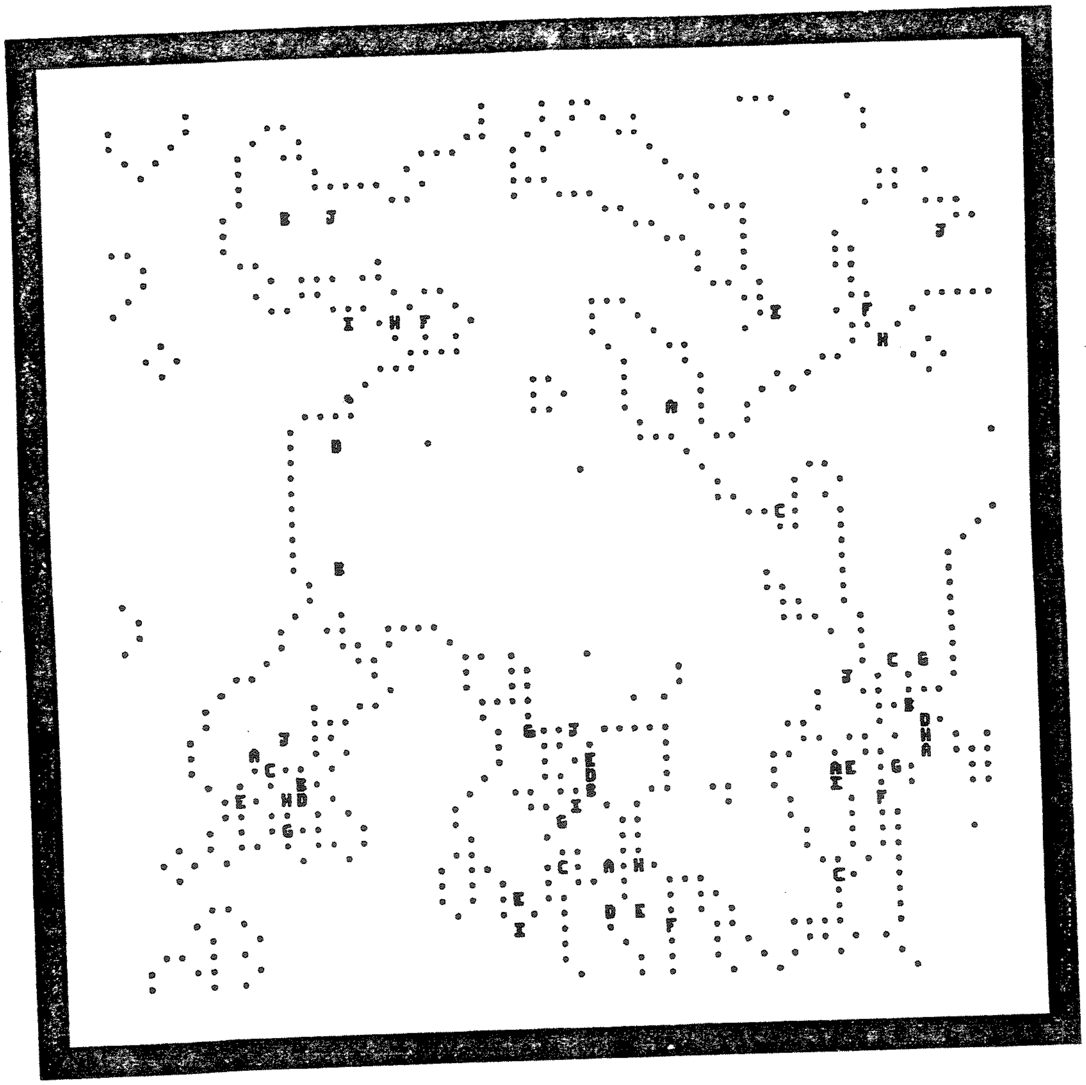

Figure 5. Edge points from the image in Figure 3, and the locations of possible matches to the points of interest from Figure 4.

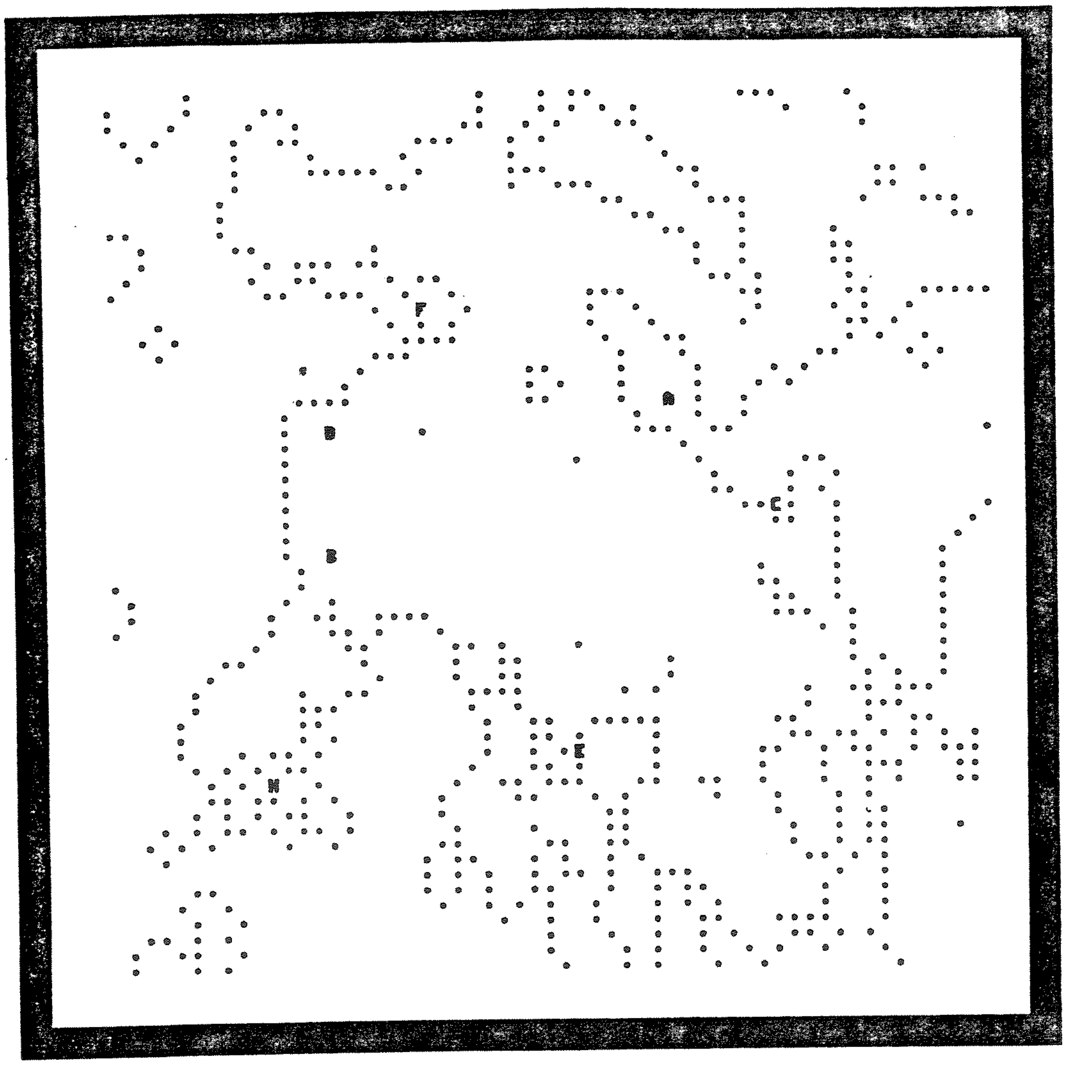

Figure 6. Resulting matched points of interest for the image in Figure 3.

References

1. J. A. Leese, C. S. Novak, and V. R. Taylor, "The determination of cloud pattern motions from geosynchronous satellite image data," Pattern Recognition, 2, 1970, 279-292.

2. J. F. Schouten, "Subjective stroboscopy and a model of visual movement detectors," in Models for the Perception of Speech and Visual Form (W. Wathen-Dunn, Ed.), MIT Press, Cambridge, Mass., 1967.

3. J. Y. Lettvin, H. R. Maturana, W. S. McCulloch, and W. H. Pitts, "What the frog's eye tells the frog's brain," Proc. IRE 47, 1959, 1940-1951.

4. D.H. Hubel and T. N. Wiesel, "Receptive fields in the cat's striate cortex," J. Physiology, 148, 1959.

5. S. Ullman, The Interpretation of Visual Motion, MIT Press, Cambridge, Mass., 1979.

6. J. W. Roach and J. K. Aggarwal, "On the ambiguity of three-dimensional analysis of a moving object from its images," WCATVI, 1979, 46-47.

7. J. Webb, "Static Analysis of moving jointed objects," Proceedings of the First Annual National Conference on Artificial Intelligence, Stanford, CA, 1980, 35-37.

8. R. Rashid, "Lights: a study in motion," Proc. Image Understanding Workshop, Los Angeles, CA., 1979, 57-68.

9. A. Rosenfeld and A. Kak, Digital Picture Processing, Academic Press, N. Y., 1976.

10. M. Duff, "CLIP 4: A large integrated circuit array parallel processor," Proc. 3rd Int. Joint Conf. on Pattern Recognition, Coronado, CA., 1976, pp. 728-733.

11. C. Rieger, "ZMOB: a mob of 256 cooperative Z80A-based micro-processors," in Proc. Image Understanding Conference, Los Angeles, CA, 1979, 25-30.

12. L. Davis, "Computer architectures for image processing" IEEE Workshop on Picture Data Description and Management, Asilomar, CA, 1980.

13. W. Eversole, D. Mayer, F. Frazec and T. Cheek Jr., "Investigation of VLSI technologies for image processing," Proc. Image Understanding Workshop, Palo Alto, CA., April 1979, 159-163.

14. W.Eversole, D. Mayer, F. Frazec, and T. Cheek Jr., "Investigation of VLSI technologies for image processing," Proc. Image Understanding Workshop, Los Angeles, CA., Nov. 1979, 10-14.

15. G. Nudd, S. Fouse, T. Nussmeier, P. Nygaard, "Development of custom-designed integrated circuits for image understanding," Proc. Image Understanding Workshop, Los Angeles, CA., Nov. 1979, 1-9.

16. D. Barnea and H. Silverman, "A class of algorithms for fast digital image registration,"IEEE Trans-Computers, C-21, 1972, 179-186.

17. N. Nilsson, Artificial Intelligence, Tioga Press, CA. 1980.

18. G. Vanderbrug and A. Rosenfeld, "Two-stage template matching," IEEE Trans-Computers, 26, 1977, 384-393.

19. G. Vanderbrug and A. Rosenfeld, "Coarse-fine template matching, "IEEE Trans-Systems, Man and Cybernetics, SMC-7, 1977, 104-107.

20. A. Klinger and C. Dyer, "Experiments in picture representation with regular decomposition, "Comp. Graphics and Image Processing, 5, 1976, 68-105.

21. T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, New York, 1978.

22. S. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing," Comp. Graphics and Image Processing, 4, 1975, 104-119.

23. J. L. Potter, "Scene segmentation by velocity measurements with a cross-shaped template," 4IJCAI, 1975, 303-308.

24. R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures,"Comm. ACM, vol. 19, 1976, 73-83.

25. D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," TR-55, Computer Sciences Dept., University of Rochester, Oct. 1979.(to appear in Pattern Recognition).

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

27. J. Sklansky, "On the Hough technique for curve detection," IEEE Trans-Computers, C-27, July 1978, 923-26.

28. L. I
ture
122.

29. W.
dyn
Rec

30. R.
of
mot
Met

31. A.
by
Cyb

32. S.
ima
lig

33. M.
Ann
197

34. J.
obj
and

28. L. Davis, "Computing the spatial structure of cellular textures," Comp. Graphics and Image Processing, 9, 1979, 111-122.

29. W. N. Martin and J. K. Aggarwal, "Computer analysis of dynamic scenes containing curvilinear figures," Pattern Recognition, vol. 11, 1979, 169-178.

30. R. M. Endlich, D. E. Wolf, D. J. Hall, and A. E. Brain, "Use of a pattern recognition technique for determining cloud motions from sequences of satellite photographs," J. Appl. Meterol., 10, 1971, 105-117.

31. A. Rosenfeld, R. Hummel, and S. W. Zucker, "Scene labelling by relaxation operations," IEEE Trans. Systems, Man and Cybernetics, vol. SMC-6, 1976, 420-433.

32. S. T. Barnard and W. B. Thompson, "Disparity analysis of images," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 4, 1980, 333-340.

33. M. Shamos, "Closest-point problems," Proc. of Sixteenth Annual Symposium on Foundations of Computer Science, ACM, 1975, 151-162.

34. J. W. Roach, and J. K. Aggarwal, "Computer tracking of objects moving in space," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 2, 1979, 127-134.