Motion Estimation of Two-dimensional Objects Based on The Straight Line Hough Transform: A New Approach [1]

Hsiang-ling Li and Chaitali Chakrabarti

Department of Electrical Engineering
Telecommunications Research Center
Arizona State University
Tempe, AZ 85287-5706
Ph: (602) 965-9516
Fax: (602) 965-8325
li@dspsun.eas.asu.edu
chaitali@asu.edu

Abstract

In this paper we propose a novel technique for estimating motion of 2-dimensional objects. The technique involves representing objects uniquely by straight-line approximations of the boundary, $\{(\theta, p)\}$, and estimating motion from shifts in the $\theta - p$ space. The straight-line approximation of the boundary is obtained by detecting dominant points during contour tracing, and then approximating the pixels between adjacent dominant points (along the contour) by a straight line using the Hough Transform. We have developed a computationally efficient way of detecting dominant points in a single pass during contour tracing. The proposed object representation enables us to determine the translation and rotation with significantly fewer computations compared to existing methods. Our method can also detect the motion parameters correctly for partially overlapped objects.

Motion estimation   Hough transform   Inverse Hough transform   Contour tracing
Dominant point   Region of support   Overlapping Objects

[1]This research was supported in part by a grant from NSF MIP-9309504.

1 Introduction

Motion estimation is the key issue in many dynamic scene analysis problems and has been an area of intensive research over the past two decades.[1,2,3,4,5,6,7,8] It has applications in a broad range of fields including video coding, data compression, autonomous navigation, and object tracking. Motion in images is recovered from a time-ordered sequence of images and is described in terms of *image positions* recorded from different views of an observed scene or in terms of *optical flow*.[7] The existing motion estimation techniques can be classified into pixel-domain based approaches and feature-domain based approaches. In pixel-domain based approaches, the amount of stray computations is very large. If instead, the object is represented by a set of features, then the number of computations can be significantly reduced. In such a case, only one translation vector and rotation angle per object would have to be transmitted, instead of displacement vectors for every group of pixels.

Most of the past work in motion estimation of two-dimensional objects has been limited to translation.[1,2] There are three major approaches for estimating two-dimensional translation: Fourier method, block matching method, and method of differentials.[3,4] Mathematical modelling techniques based on operator formulation and Kalman filter have also been used for two dimensional motion estimation of noisy images.[5,6] All these approaches are pixel-domain based and require a significant amount of computation. In recent years, several researchers have used feature-domain based approaches to estimate motion parameters. One such method uses the Hough Transform to estimate rotation and translation of two-dimensional objects.[11,12]

The Hough transform (HT) is a well-known method for the detection of parametric curves. However, high computational cost poses severe restrictions on its applicability to large images. One effective approach of overcoming this drawback is parameter space decomposition,[10] which enables the transform to be expressed as the sum of two terms, namely, the translation term and the intrinsic term.[12] Such a decomposition makes the estimation of the motion parameters easy and has been adopted in the procedures of Casasent et al.[11] and Pao et al.[12] The work proposed by Casasent et al.[11] avoids the problems of high dimensionality and extensive preprocessing of the generalized HT but is limited to analytic curves such as circles, ellipses, and parabolas. More recently, Pao et al.[12] have generalized and extended this method for arbitary shapes by computing the perpendicular distances between pairs of parallel tangent lines as the signature of a shape. The rotation and translation is then determined by utilizing 1-D correlations of the reference signature with the test signature. The main drawback of this approach[12] is that it is computationally intensive since the object is represented by a large number of straight lines, and thus a proportionately large number of computations have to be done to determine the motion parameters.

In this paper, we propose a novel feature-domain motion estimation technique based on the straight-line Hough Transform (SLHT). We achieve a significant reduction in the number of computations compared to Casasent et al[11] and Pao et al[12] by representing the object more compactly. Specifically, we represent the object by a *small* set of straight lines, where each straight line approximates the contour between two points of high curvature or dominant points. Each straight line is represented by a (θ, p), where p[2] is the length of the normal and θ is the angle between the normal and the $x-$axis. Thus, the object is represented by a *small* set of peaks in the $\theta - p$ space. Such an object representation is robust and less sensitive to discretization errors. In order to reduce the overhead in detecting dominant points, we develop a computationally efficient way of detecting these points in a single pass during contour tracing. The straight-line approximation of the object allows us to efficiently estimate translation and rotation from shifts in the $\theta - p$ space. The proposed method of motion estimation can also be extended to estimate rotation and translation correctly for partially overlapping objects.

The rest of this paper is organized as follows. Section 2 gives the overview of the proposed

[2] $p = x cos\theta + y sin\theta$, where (x, y) is a point on the line.

method. Section 3 describes the procedure to detect dominant points. Section 4 describes the estimation of rotation and translation based on the SLHT with examples. Section 5 discusses the detection of rotation and translation when several objects overlap. Section 6 derives the computational complexity of our proposed method. This paper is concluded in Section 7.

2 Overview

In order to reduce the number of computations in motion estimation, the boundary of an object needs to be represented compactly. In the existing procedures, the boundary of an object is approximated by tangent lines through almost every boundary pixel. Such a procedure requires an enormous amount of computation. Moreover, a threshold value in the $\theta - p$ space is required to reduce the number of spurious peaks. In this paper, we propose a method to represent an object compactly and uniquely. We use dominant points in the representation of an object since it is well known that information on a 2-D shape is concentrated at the dominant points.[13,14,15,16,17,18,19,20,21,22]

The first step in our procedure is to trace the contour and detect the dominant points. Next, the contour is broken into groups of edge pixels with dominant points serving as the end points of each group. Then, each group of edge pixels is approximated by a straight line. The reason why we do not rely solely on dominant points to represent the object is that discretization errors can cause displacements in the positions of the dominant points. This is especially true when objects are rotated. By computing the Hough Transform of pixels between two adjacent (along the contour) dominant points, an object is represented more robustly.

Once an object is represented uniquely by straight-line approximation of the boundary or by a set of (θ, p)s, we estimate the motion parameters of the object by computing the shifts in the $\theta - p$ space. Let $\{(\theta, p)\}_t$ and $\{(\hat{\theta}, \hat{p})\}_{t+1}$ be the representations of the object at times

t (reference object) and $t+1$ (test object), respectively. We first determine the rotation angle ϕ by computing the votes on $\Delta\theta$s, the differences in the θ values of the object at times t and $t+1$. Then we compute the approximate translation vector from the displacement of the centroid of the polygon whose vertices are the dominant points. We then refine the approximate translation vector by computing the inverse Hough Transform on $\{(\theta, \Delta p)\}$ where $\Delta p = \hat{p}(\theta+\phi) - p_0(\theta)$, and $p_0(\theta)$ is the intrinsic term obtained from $p(\theta)$ of the reference object. The block diagram of the motion estimation system is shown in Figure 1.

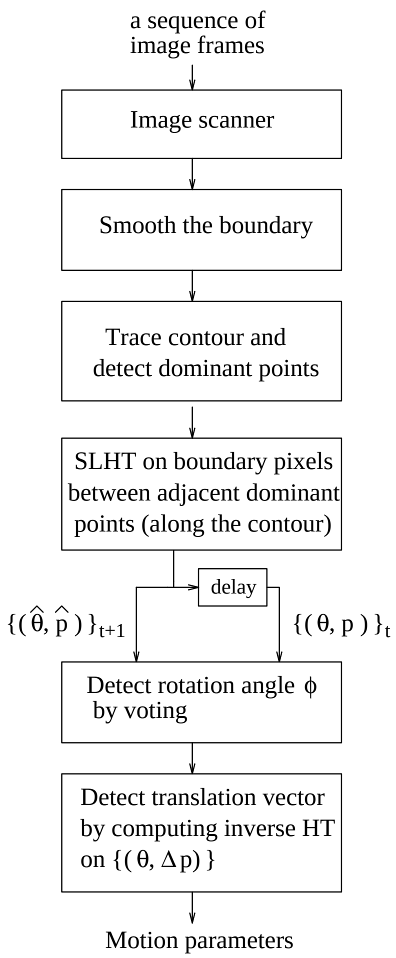


Figure 1: Block diagram of the motion estimation system.

3 Object Representation

In our method, an object is represented by straight line approximations of the boundary, where each straight line approximates the contour segment between two dominant points. Thus, one of the most crucial procedures in the proposed method is the detection of dominant points. Correct detection of dominant points relies primarily on the precise determination of the region of support.[14] This is because the boundary of a planar object consists of multiple size features, and hence the region of support and the corresponding scale or smoothing parameters are different for each boundary point. In the proposed method, the local properties of the curve are used to determine the region of support.

The procedure for determining the dominant points consist of two steps. In the first step, the contour is traced so that the chain code and curvature of every edge pixel can be evaluated. In the next step, the region of support for each dominant point is determined from the local properties of the contour. The pixel having the absolute maximum curvature within a region of support is then identified as the dominant point in that region. We describe these steps in more details next. Note that the proposed method detects dominant points in a single pass and is significantly less computationally intensive compared to existing methods that are based on convolving with Gaussian windows of variable widths.

3.1 Contour Tracing

In this procedure, the image frame is partitioned into several subimages, and the contours in each subimage are generated in parallel. Next, contour integration is done to link the contours in each subimage. Such a procedure has the advantage of being easily mapped into a parallel processing environment. We implement the contour tracing module with a state machine having 68 states where each state corresponds to a distribution of pixels in a 3×3 window centered around the current edge pixel. The procedure is summarized below:

Step 1: Smoothing the boundary and removing noise pixels

Discrete boundary representation and quantization errors can cause introduction of false or noise pixels along a boundary. The first 40 states in the state machine are used to remove these noise pixels and smooth the boundary.

Step 2: Contour tracing

In each subimage we first detect the starting pixels of all the contours. Then, we start tracing the contour in the counterclockwise direction from one of the starting pixels. 28 states in the state machine are used to perform the above task. If a match occurs on any of the 28 states, the next edge pixel (which is the center pixel of the next 3×3 window), and the chain code of the current edge pixel relative to the previous edge pixel (in the contour) are both determined. Please refer to Figure 2 for the details. The tracing of a particular contour stops at the boundary of the subimage. The same procedure is used to trace all the contours in a subimage.

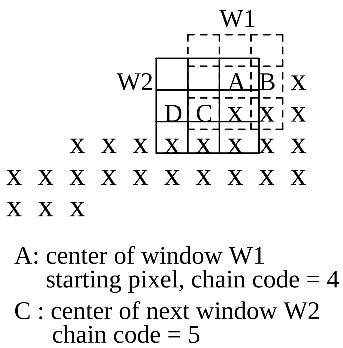


Figure 2: Contour tracing by a 3×3 window.

During contour tracing, the Freeman chain code and the curvature of every edge pixel on the contour are calculated. This information is used to detect the dominant points within each region of support.

3.2 Detection of Dominant Points

Most of the existing dominant point detectors either are single-scale algorithms that only work well for an object of similar size features, or use computationally intensive scale-space mapping(18,19,20,21,22) for multiple-scale curves. None of the above methods are appropriate for motion estimation or other procedures which demand small latency. Here, we propose a new and efficient algorithm that detects the region of support in a single pass during contour tracing. The algorithm uses the local properties of the curve to determine the size of the region of support.

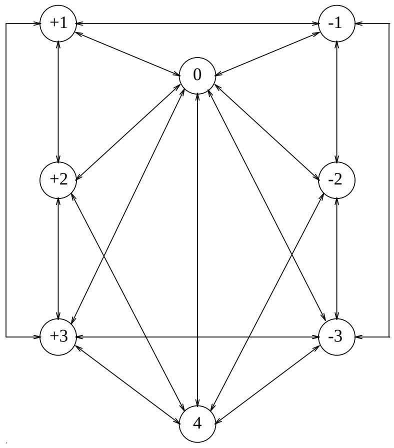


Figure 3: The state diagram for determining the region of support.

In the proposed algorithm for detection of the region of support, we need three parameters n_1, n_2, and n_3. If the number of consecutive pixels with the same chain code is $< n_1$, the system is in transition and the pixels do not contribute to the region of support. Such a situation occurs due to discrete boundary representation. n_2 specifies the minimum number of valid pixels to represent one side of a large corner, and n_3 specifies the minimum number of valid pixels to complete a small corner or a convex/concave region. These three parameters play a role similar to that of σ in the Gausian kernel used in scale-space mapping.

Figure 3 describes the state diagram that is used to determine the region of support during contour tracing. There are eight states, with state(0) being the reference state for detecting the region of support. During contour tracing, the system moves from one state to another whenever two consecutive edge pixels (in the contour) have different chain codes. From state(j), the system goes to state(i) if the difference between the chain code of the current edge pixel and the chain code of the pixel when the system was in state(0), is i and $-3 \leq i \leq 4$. Since the chain codes are represented by Freeman code, adjustments have to be done so that the state label, i is in the specified range. The following rule is utilized,

$$\text{if } 5 \leq i \leq 7, \text{ then } i = i - 8.$$
$$\text{if } -7 \leq i \leq -4, \text{ then } i = i + 8.$$

The state transitions give us valuable information about the nature of the curve that is being traced. We can determine whether the system is tracing a sharp corner or a large corner or a convex/concave region, from the state transitions. For instance, if the system makes transitions starting from state(i) and ending at state(j) and it was in either of state(i) and state(j) for at least n_2 time units, and if $|i - j| = 1$ or 2, a large corner is detected. On the other hand, if $|i - j| = 3$ or 4, a sharp corner is detected.

We next define the following terms before describing the algorithm in detail.

- **Duration of a state:** The number of consecutive pixels with a common chain code corresponding to a state is defined as the duration of a state. The duration of a state is reset to 0 whenever a different state becomes active.

- **Effective pixels, states and duration:** If the duration of a state is $\geq n_1$, The duration is said to be effective. The state and corresponding pixels are also said to be effective.

- **Status:** The chain code corresponding to a state is defined as the status of a state.

- **Count:** The total number of effective pixels with a common chain code is defined as the count of a state. The count of each state is reset to 0 initially, and can only be increased by an amount equal to the effective duration. Otherwise, the count of a state remains the same as before the duration.

- **Front pointer (FP):** The position of the first effective pixel of the first state in a region of support is stored as the front pointer of the corresponding region of support. After the count of the first state reaches n_2, the front pointer moves such that the number of effective pixels between the FP and the last effective pixel of the first state remains at n_2 (in order to reduce the size of the region of support).

- **Rear pointer (RP):** When the count of the last state in a region of support reaches n_2 (for the case of a large corner) or n_3 (for the case of a sharp corner), the position of the n_2th effective pixel or the n_3th effective pixel is stored as the rear pointer of the corresponding region of support. Such a procedure ensures reduction in the size of the region of support.

Description of the algorithm:

Let state(0) correspond to the chain code of the first pixel. The first few transitions (before the detection of an effective state) are between the states 0 through 4. Once an effective state is detected, its status, and count ($\geq n_1$), are shifted to state(0) and state(0) is classified as being effective. We next describe the conditions that are required for the detection of (i) obvious corner and (ii) concave or convex region. After the detection of a corner, the following two steps are computed.

• All the pixels between the front pointer and the rear pointer constitute a region of support.

• The status, and count of the last state in a region of support are shifted to state(0). The statuses and counts of all other states are reset to 0. The contour tracing procedure then resumes for the next boundary pixel.

Case 1: Obvious corner

• Assume that the count of state(i), $-2 \leq i \leq 2$, reaches n_2 first. The position of the first effective pixel of state(i) is stored as the front pointer.

• When the count of another state, state(j), $-2 \leq j \leq 2$, also reaches n_2 and $|i - j| = 1$ or 2, we have detected an obvious corner. The position of the n_2th effective pixel of state(j) is stored as the rear pointer.

Example 1: Consider the boundary of a 2-dimensional object as shown in Figure 4. Let n_1=2, n_2=6 and pixel 0 be the starting pixel (chain code = 4). Figure 5 shows the transition of states for this example. Since pixel 5 is in transition, the status and count of state(0) remain the same. The counts of state(0) and state(2) are both equal to 6 ($\geq n_2$), and the absolute difference between the two state labels is 2. Hence, we have detected an obvious corner. In addition, the region of support consists of pixels 2 through 18 since FP is 2 and RP is 18. The status, and count of state(2) are shifted to state(0), and the contour tracing starts from the next boundary pixel, i.e. pixel 19. □

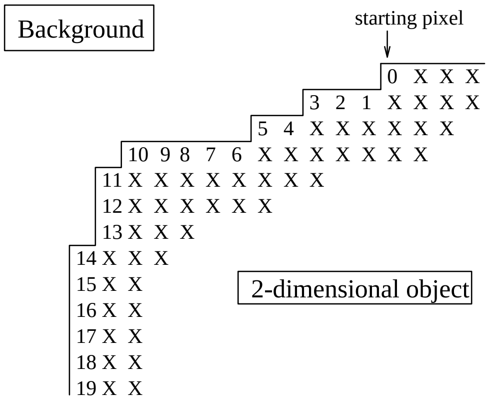


Figure 4: Dominant point detection: an example of an obvious corner.

Case 2: Convex or concave region

• In this case, we may or may not detect the first effective side of an obvious corner, i.e. one of the counts of the states whose labels are in the range from -2 to 2, may be $\geq n_2$.

• A convex or concave region is said to have been detected if before we are able to detect the second effective side of an obvious corner,

(i) either state(j), $-2 \leq j \leq 2$, becomes effective, the count of state(j) reaches n_3, and $|i-j| = 3$ or 4, where i is $-2 \leq i \leq 2$. The position of the first effective pixel of state(i) is stored as the front pointer and the position of the n_3th effective pixel of state(j) is stored as the rear pointer.

(ii) or state(j), where j is among {-3, 3, 4}, becomes effective and the count of state(j) reaches n_3. The position of the first effective pixel of state(0) is stored as the front pointer and the position of the n_3th effective pixel of state(j) is stored as the rear pointer.

Example 2: Let the boundary shown in Figure 6 be the continuation of the boundary in Example 1, n_1=2, n_2=6, and n_3=3. While tracing the contour pixels 19 through 34, the

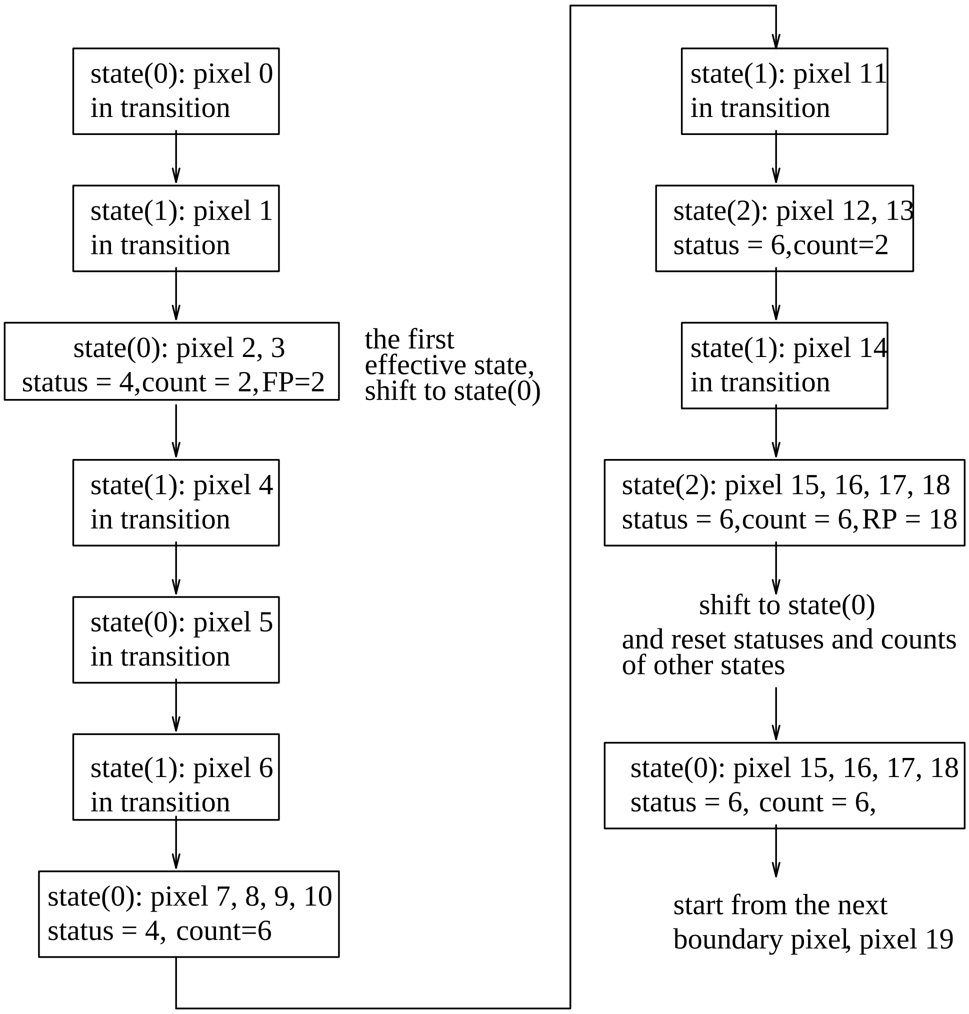


Figure 5: Transition of states in Example 1 during dominant point detection.

state machine undergoes transitions among the following states: state(0), state(1), state(2), state(1), state(0), state(-1), and state(-2). The count of state(-2) is 3 ($= n_3$), and the absolute difference between the labels of state(-2) and state(1) (or state(2)) is 3 (or 4). Hence, we have detected a sharp corner (a concave region). In addition, the region of support consists of pixels 22 through 34 since FP is 22 and RP is 34. The status, and count of state(-2) are shifted to state(0), and the contour tracing starts from the next boundary pixel, i.e. pixel 35. □

Example 3: Let the boundary shown in Figure 7 be the continuation of the boundary in Example 2, n_1=2, n_2=6, and n_3=3. While tracing the contour pixels 35 through 47, the

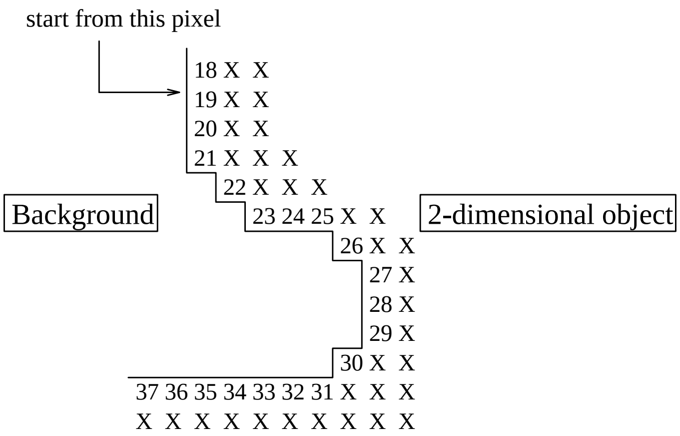


Figure 6: Dominant point detection: an example of a concave region or a sharp corner .

state machine undergoes transitions among the following states, state(0), state(1), state(2), state(3), and state(4). The count of state(4) is 3 ($= n_3$), and the absolute difference between the labels of state(0) and state(4) is 4. Hence, we have detected a sharp corner (a convex region). In addition, the region of support consists of the pixels from pixel 33 to pixel 47 since FP is 33 and RP is 47. The status, and count of state(4) are shifted to state(0), and the contour tracing starts from the next boundary pixel, i.e. pixel 48. It should be noted that original FP is shifted forward by 1 since the count of state(0) is equal to 7. □

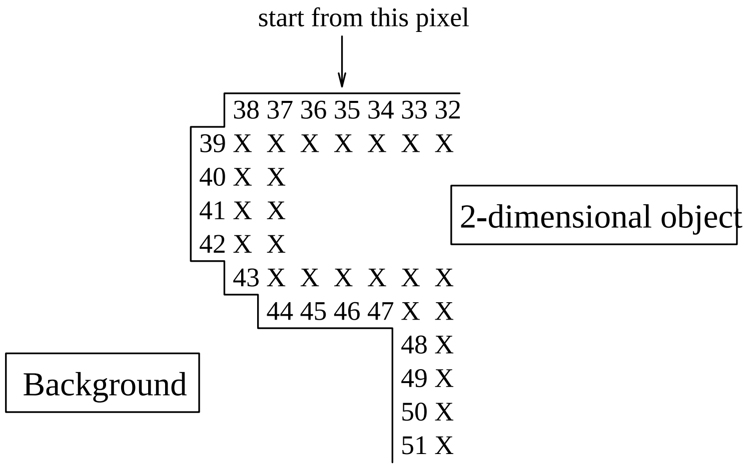


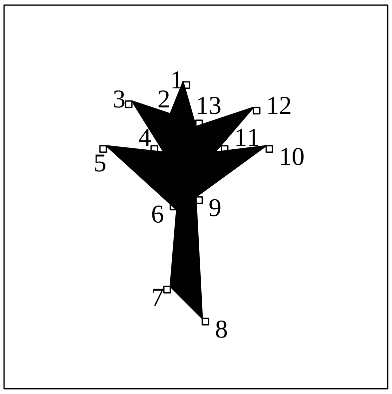
Figure 7: Dominant point detection: an example of a convex region or a sharp corner.

After the region of support has been determined, the curvature of every pixel in the region is computed using the k-curvature method.[14] The pixel having the absolute maximum curvature within a region of support is the dominant point in that region. It is known that if the region of support is determined correctly, the actual value of k does not cause any significant difference.[14]

3.3 Straight Line Approximation of The Boundary

Once the dominant points are detected, we break the contour into groups of edge pixels with dominant points serving as the end points of each group. Then, each group of edge pixels is approximated by a straight line segment (θ, p) using the SLHT. To reduce the number of computations, the $(\tilde{\theta}, \tilde{p})$ of a straight line connecting the end points of each group is computed first. Then, the Hough Transform of the pixels between two adjacent dominant points (along the contour) is computed around the initially estimated $\tilde{\theta}$ value. This results in a significant reduction in the size of the parameter space. It should be noted that no thresholding is required here, since each group of pixels gives rise to a single peak for one straight line.

Example 4: The dominant points detected by the proposed method for three 2-dimensional objects are shown in Figure 8 (a), (b) and (c).



(a)

Figure 8: Dominant points labelled in (a) a leaf (b) a wrench (c) a wire-cutter.

4 Detection of Translation and Rotation

In this section, we show how the inverse Hough Transform can be used to detect rotation and translation of objects that are represented by straight-line approximations of the boundary. The most significant contribution of our method is that the parameter space and the number of computations are much smaller than the techniques proposed in reference (11) and (12).

4.1 SLHT of a Line Segment

The straight line Hough Transform (SLHT) can be expressed as the sum of two terms, namely, the translation term and the intrinsic term (the parameter space decomposition[(10)]). This formulation allows the translation parameters to be easily separated, and rotation to be determined from circular shifts in the $\theta - p$ space. Let the line segment $l = (\theta, p)$ be a straight line approximating part of the boundary of an object C centered at (x_0, y_0). The value of θ is confined to the range 0^o to 180^o, and p is either positive or negative.

$$p(\theta) = p_0(\theta) + t_0 cos(\theta - \alpha_0), \tag{1}$$

where $p_0(\theta)$ is the intrinsic term (the SLHT of that line segment when C is located at the origin), $t_0 = \sqrt{x_0^2 + y_0^2}$, and $\alpha_0 = tan^{-1}(y_0/x_0)$. If the shape C is rotated by an angle ϕ ($< 180^o$ to avoid confusion) in the counterclockwise direction and the center of the shape C is translated to (x^*, y^*), then the point (θ, p) is mapped to $(\hat{\theta}, \hat{p})$, where

$$\begin{aligned} \hat{\theta} &= (\theta + \phi) \bmod 180^o, & (2) \\ \hat{p}(\hat{\theta}) &= p_0(\theta) + t_0 cos(\theta + \phi - \alpha_0) + t_1 cos(\theta + \phi - \alpha_1), & (3) \end{aligned}$$

where $t_1 = \sqrt{(x^* - x_0)^2 + (y^* - y_0)^2}$ and $\alpha_1 = tan^{-1}(\frac{y^* - y_0}{x^* - x_0})$. Figure 9 gives a pictorial description of the above formulation.

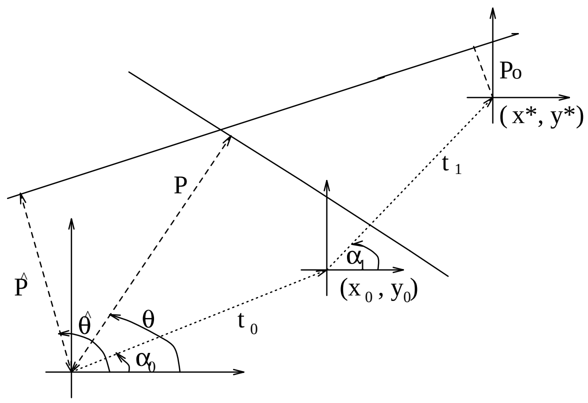


Figure 9: SLHT of a line segment (θ, p) after translation by $(x^* - x_0, y^* - y_0)$ and rotation by ϕ.

4.2 Calculation of the Rotation Angle

The above derivation (refer to equation 2) shows that whenever an object is rotated by ϕ ($< 180^o$), $\hat{\theta}$ of the rotated object is obtained by adding ϕ to the θ of the original object. The procedure for calculating ϕ consists of computing the votes corresponding to the differences in the θ values of the object at t (reference object) and $t+1$ (test object), and assigning ϕ to be the angular difference with the highest vote. The procedure for detecting the rotation angle is described below.

/$*$ h[t]: accumulated scores of the differences between θs $*$/
for i=1 to n *do*/$*$ θ_i: θs of the object at t $*$/
    for j=1 to n *do*/$*$ θ_j: θs of the object at $t+1$ $*$/
        if(($\theta_j - \theta_i$) > 0.0)
            $\tilde{\phi} = \theta_j - \theta_i$;
        else $\tilde{\phi} = \theta_j - \theta_i + \pi$;
        $\tilde{\phi}$ is quantized to an appropriate value, k ;
        h[k]++;
detect peak in score h[t];

The difference between two corresponding angles θ and $\hat{\theta}$ may not be exactly equal to ϕ, i.e. $\hat{\theta} \cong (\theta + \phi) \bmod 180^o$ due to errors caused by the discretization of the image and quantization. Hence, the detected peak does not always correspond to the correct rotation angle. A better estimate of the rotation angle ϕ can be obtained by using the average of the angles around $\tilde{\phi}$, i.e. in the range from $\tilde{\phi}-\Delta\tilde{\phi}$ to $\tilde{\phi}+\Delta\tilde{\phi}$. The value of $\Delta\tilde{\phi}$ is determined by the properties of the boundary. A boundary with higher curvature requires a larger value of $\Delta\tilde{\phi}$.

4.3 Calculation of the Translation Vector

After the rotation angle is determined, a one-on-one correspondence is established between $\{(\theta,p)\}_t$(reference object) and $\{(\hat{\theta},\hat{p})\}_{t+1}$ (test object). This correspondence is required for the calculation of the translation vector. The procedure is described below.

First, the intrinsic term, $p_0(\theta)$, is calculated for each θ by subtracting the corresponding translation term, $T(\theta) = t_0 cos(\theta - \alpha_0)$ from $p(\theta)$ of the reference object (refer to equation 1). Next, $\Delta p(\theta) = \hat{p}(\hat{\theta}) - p_0(\theta)$ is calculated for each θ (refer to equation 3). If $\theta + \phi > \pi$, the corresponding $\hat{p}(\hat{\theta})$ is made negative during the computation of $\Delta p(\theta)$. Since Δp can be rearranged as

$$\Delta p(\theta) = x^* cos(\theta + \phi) + y^* sin(\theta + \phi),$$

we can compute x^* and y^* by performing an inverse Hough Transform on $\{\theta, \Delta p\}$. The time to locate the peak in the $x-y$ plane can be reduced by first approximating x^* and y^* and then refining the search in a smaller area. The approximate values of x^* and y^* are the centroid of the polygonal approximation of the test object, whose vertices are the dominant points.

4.4 Experimental Results

In this section, two examples are presented to illustrate the proposed method for motion estimation.

Example 5:

Figure 10 (a) and (b) show the reference object $C1$ and the test object $C2$ (after translation by $(120, 50)$ and rotation by 90 degrees). The ten dominant points detected for $C1$ and $C2$

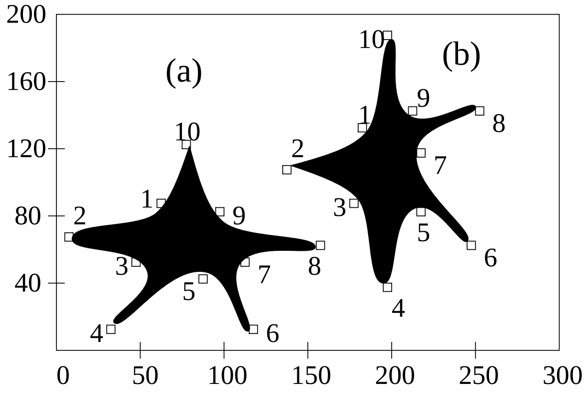


Figure 10: (a) C1: reference object at time t (b) C2: test object at time $t+1$

are indexed on the contours (see Figure 10 (a) and (b)). The straight-line approximation of the contour between two dominant points is computed with quantization intervals of $\Delta p = 1.0$, and $\Delta\theta = 1^o$. The $\{(\theta, p)\}$ and $\{(\hat{\theta}, \hat{p})\}$ for $C1$ and $C2$ are listed in Figure 11.

	Reference object C1		Test object C2		translation term T	p_o	Δp
	p	θ	$\hat{p}$	$\hat{\theta}$			
1	57	113	53	110	24.2	32.8	-234.8
2	63	77	- 162	66	76.6	-13.6	-144.4
3	- 37	158	- 202	16	-51.5	14.5	-207.5
4	- 39	115	- 158	163	20.8	-59.8	-188.2
5	105	29	- 193	65	99	6	-135
6	112	4	- 248	27	83.86	28.14	89.86
7	28	102	- 129	120	42.26	-14.26	-215.74
8	103	71	118	95	82.9	20.1	-153.1
9	111	17	- 230	8	94	17	36
10	- 30	156	- 133	159	-48.5	18.5	-180.5

Figure 11: Example 5: $\{(\theta, p)\}$ values of C1, $\{(\hat{\theta}, \hat{p})\}$ values of C2, $T(\theta)$, $p_0(\theta)$ and $\Delta p(\theta)$

The estimated rotation angle $\tilde{\phi}$ corresponding to the highest vote is 91^o. The refined rotation angle $\phi = 89.5^o$ is obtained by averaging the angles around $\tilde{\phi}$ with the maximum deviation $\Delta\tilde{\phi} = 4^o$. Based on this rotation angle, a one-on-one correspondence between $\{(\theta, p)\}$ and $\{(\hat{\theta}, \hat{p})\}$ is established.

The centroid of the polygon approximation of the reference object $C1$ is $(80, 60)$ and the translation term is thus equal to $T(\theta) = 100cos(\theta - 37^o)$. The intrinsic term, $p_0(\theta)$ for each θ is obtained by subtracting the corresponding translation term, $T(\theta)$ from $p(\theta)$ of the reference object $C1$. Then, $\Delta p(\theta)$ for each θ is computed by subtracting $p_0(\theta)$ from corresponding $\hat{p}(\hat{\theta})$. The results of the above calculations are listed in Figure 11.

The centroid of the polygon approximation of the test object $C2$ is $(x^*, y^*) = (200, 110)$ and is refined by computing the inverse Hough transform of $\{(\theta, \Delta p)\}$. By doing such a coarse to fine search, the voting in the $x - y$ plane is computed in a small area (20×20 pixels) around $(200, 110)$. The refined (x^*, y^*) corresponding to the highest vote count is equal to $(201.5, 107.5)$ and the translation vector is $(121.5, 47.5)$.

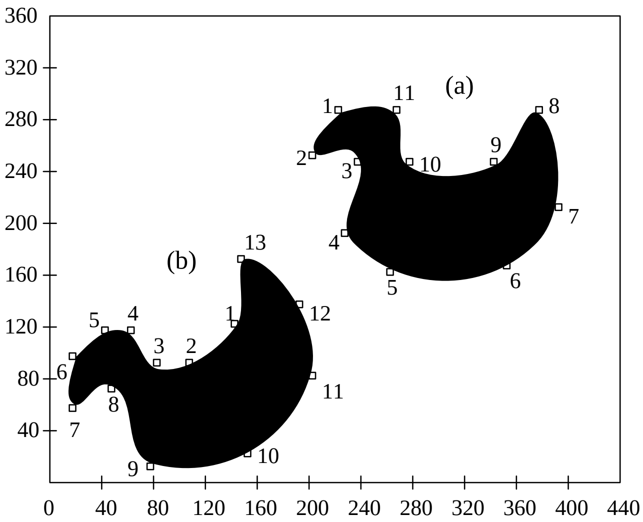


Figure 12: (a) C3: reference object at time t (b) C4: test object at time $t+1$

Example 6:

Figure 12 (a) and (b) show the reference object $C3$ and the test object $C4$ (after translation by $(-186, -144)$ and rotation by 30 degrees). $C3$ has 11 dominant points while $C4$ has 13 dominant points. The difference in the numbers of dominant points is due to discretization and quantization errors. The $\{(\theta, p)\}$ and $\{(\hat{\theta}, \hat{p})\}$ for $C3$ and $C4$ are listed in Figure 13.

The estimated rotation angle $\tilde{\phi}$ corresponding to the highest vote is 33^o and the refined rotation angle ϕ is equal to 31.5^o. The maximum deviation $\Delta\tilde{\phi} = 5^o$. The results of the computed translation terms, $p(\theta)$s, and $\Delta p(\theta)$s are listed in Figure 13. The approximate (x^*, y^*) is equal to $(126, 95)$, and the refined (x^*, y^*) is equal to $(124, 100)$. The translation vector is then $(-184, -127)$. Note that even though the numbers of dominant points detected for the reference and test objects are not equal, the results are quite satisfactory.

	Reference object C3 p	θ	Test object C4 $\hat{p}$	$\hat{\theta}$	translation term T	p_0	Δp
1	0	146	-10	120	-133.8	133.8	-142.8
2	316	74	91	87	293.6	22.4	32.6
3	-161	168	108	31	-256	95	-168
4	299	49	121	85	365.9	-66.9	74.9
5	134	94	66	127	195.2	-612	-33.8
6	-37	125	-9	0	1.3	-38.3	244.3
7	398	1	55	111	311.6	86.4	148.6
8	-151	146	-73	21	-133.8	-17.2	-117.8
9	244	87	8	92	233	11	-21
10	298	6	-95	135	328.9	-30.9	138.9
11	281	92	-206	2	206.3	74.7	-8.7
12			235	45			
13			-135	175			

Figure 13: Example 6: $\{(\theta, p)\}$ values of C3, $\{(\hat{\theta}, \hat{p})\}$ values of C4, $T(\theta)$, $p_0(\theta)$ and $\Delta p(\theta)$

The summary of the motion estimation results using the proposed method is listed in Figure 14. The table shows that our method tends to have larger errors in the estimated motion parameters for objects having long and high-curvatured boundaries, e.g. "wire-cutter" and "duck". On the other hand, our method estimates motion parameters quite correctly for objects having smooth or straight boundaries or sharp corners, e.g. leaf, wrench, star fish, C5 and C6 (Example 8 is described in Section 5.3).

	actual rotation angle	estimated rotation angle	error	actual translation vector	estimated translation vector	error vector
example 4(a) : leaf	15^o	14.8^o	0.2	(-212 , 28)	(-215.5 , 29)	(3.5 , -1)
example 4(b) : wrench	20^o	21^o	1	(-127 , -166)	(-128 , -173)	(1 , 7)
example 4(c) : wire-cutter	25^o	28.5^o	3.5	(158 , 89)	(154.5 , 72)	(3.5 , 17)
example 5 : star fish	90^o	89.5^o	0.5	(120 , 50)	(121.5 , 47.5)	(-1.5 , 2.5)
example 6 : duck	30^o	31.5^o	1.5	(-186 , -144)	(-184 , -127)	(-2 , -17)
example 8 : overlapping objects	20^o (C5) 15^o (C6)	19.5^o(C5) 15.2^o(C6)	0.5 (C5) 0.2 (C6)	(-22,126) (C5) (-213,125) (C6)	(-26,130.5) (C5) (-210,126) (C6)	(4 , -4.5) (C5) (-3 , -1) (C6)

Figure 14: The summary of the motion estimation results using the proposed method.

5 Motion Estimation for Overlapping Objects

The problem of identifying partially overlapping objects from an image has drawn much research interest in the area of computer vision.[23,24] This problem has significant importance in industrial environment as well as in military applications. In this section, we demonstrate that the motion estimation method described in Section 4 can also be used to estimate rotation and translation correctly for overlapping objects. The procedure includes recognizing which candidate objects are involved in the occlusion using the principle of maximum likelihood and then determining motion parameters of involved candidate objects using the information obtained from the non-overlapping parts.

5.1 Object Recognition

Let a 2-dimensional image of an object be uniquely represented by $\{(\theta, p)\}$s, where the θs are indexed in the counterclockwise direction. Let the absolute difference between two consecutive θ values be denoted as $\delta\theta_i$, where $\delta\theta_i = |\theta_j - \theta_i|$, $j = (i+1) \bmod n$, $i = 1, \ldots, n$, and n is the number of dominant points of the planar object. Obviously, $\delta\theta_i$ is not effected by translation and scaling. Next, we consider the effect of rotation. Let the object be rotated by ϕ ($< 180^o$ to avoid confusion) in the counterclockwise direction and let the rotated object be represented by $\{(\hat{\theta}, p)\}$s, where $\hat{\theta} = (\theta + \phi) \bmod 180^o$, and the $\hat{\theta}$s are indexed in the counterclockwise direction. Let $\delta\hat{\theta}_i$ be the absolute difference between two consecutive $\hat{\theta}$ values, where $\delta\hat{\theta}_i = |\hat{\theta}_j - \hat{\theta}_i|$, $j = (i+1) \bmod n$, and $i = 1, \ldots, n$. Then, $\delta\theta_i$ and $\delta\hat{\theta}_i$ are related as follows:

$$\delta\theta_i = \delta\hat{\theta}_i \qquad \text{or} \qquad \delta\theta_i + \delta\hat{\theta}_i = 180^o.$$

This leads us to conclude that $\delta\theta_i$, $i = 1, \ldots, n$ is invariant to translation, rotation and scaling and can thus be used to identify the candidate objects in a compound object.

5.2 Identification of Candidate Objects

Let $\{\delta\hat{\theta}\}_N$ be the set of $|N|$ $\delta\hat{\theta}$ values corresponding to the compound (new) object and $\{\delta\theta\}_{Ci}$ be the set of $|Ci|$ $\delta\theta$ values corresponding to the ith candidate object. The procedure to determine which part of the new object belongs to some candidate object is described below.

```
for i = 1 to C do (Repeat for every candidate object)
j = 1;
while (j is inside {δθ̂}_N)
    for each δθ_k in {δθ}_Ci do
        if (δθ_k = δθ̂_j or δθ_k + δθ̂_j = 180°) then add k to LIST(j);
    if LIST(j) is empty, then j++;
    else
        {
            r = the first index in LIST(j);
            p = (r+1) mod |Ci|;
            q = (j+1) mod |N|;
            while (r is inside LIST(j))
                if (δθ_p = δθ̂_q or δθ_p + δθ̂_q = 180°)
                    add an edge between q-1 and q;
                    p = (p+1) mod |Ci|;
                    q = (q+1) mod |N|;
                else
                    r = the next index in LIST(j);
                    p = (r+1) mod |Ci|;
                    q = (j+1) mod |N|;
            store the path (a group of δθ̂s connected together by edges) in LIST_Ci(j);
            let j be the index behind the last element of LIST_Ci(j);
        }
```

The path stored in $LIST_{Ci}(j)$ represents the longest path in the compound object that starts with index j and belongs to the ith candidate object. We determine the candidate objects that are involved in the occlusion using the principle of maximum likelihood. For instance, let $LIST_{C0}(2)$, and $LIST_{C1}(4)$ be subsets of $LIST_{C2}(1)$. Then, we should remove $LIST_{C0}(2)$, and $LIST_{C1}(4)$, and conclude that the features of the compound object described by $LIST_{C2}(1)$ belongs to the 2nd candidate object. Once we identify the candidate objects, rotation and translation parameters are estimated using the information obtained from the nonoverlapping parts.

5.3 Experimental Results

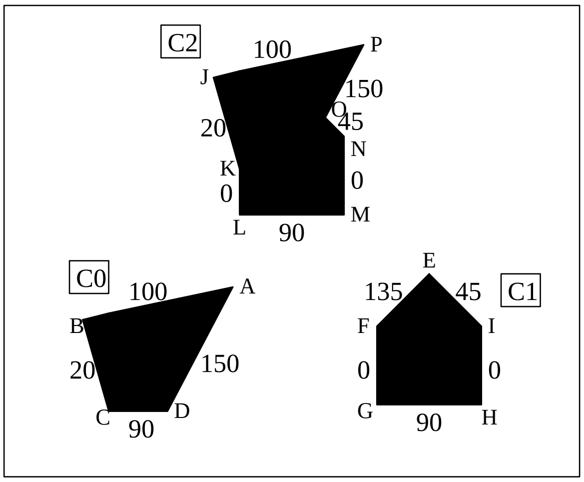


Figure 15: $C0$ and $C1$ are candidate objects and $C2$ is the test object.

Example 7: $C0$ and $C1$ are candidate objects and $C2$ is the test object obtained by overlapping $C0$ and $C1$ in Figure 15. The dominant points of each object are indicated on their contours by letters (A, B, ...). The θ values of each group of edge pixels are also shown on the contour. The absolute difference between two adjacent θ values (denoted as $\delta\theta$) is computed for every object. The $\{\delta\theta\}_{C0}$, $\{\delta\theta\}_{C1}$, and $\{\delta\hat{\theta}\}_{C2}$ for the three objects are shown below;

$$\{\delta\theta\}_{C0} = \{70, 60, 50, 80\},$$
$$\{\delta\theta\}_{C1} = \{135, 90, 90, 45, 90\},$$
$$\{\delta\hat{\theta}\}_{C2} = \{80, 20, 90, 90, 45, 105, 50\}.$$

The results of comparing $\{\delta\theta\}_{C0}$ and $\{\delta\theta\}_{C1}$ with $\{\delta\hat{\theta}\}_{C2}$ are stored in $LIST_{C0}(7)$, and $LIST_{C1}(3)$, respectively. Here,

$$LIST_{C0}(7) = \{50, 80\},$$

$$LIST_{C1}(3) = \{90, 90, 45\}.$$

Since these two sets are disjoint, we conclude that the dominant points of $C2$ indexed as P and J belong to $C0$, and the dominant points indexed as L, M and N belong to $C1$. $\square$

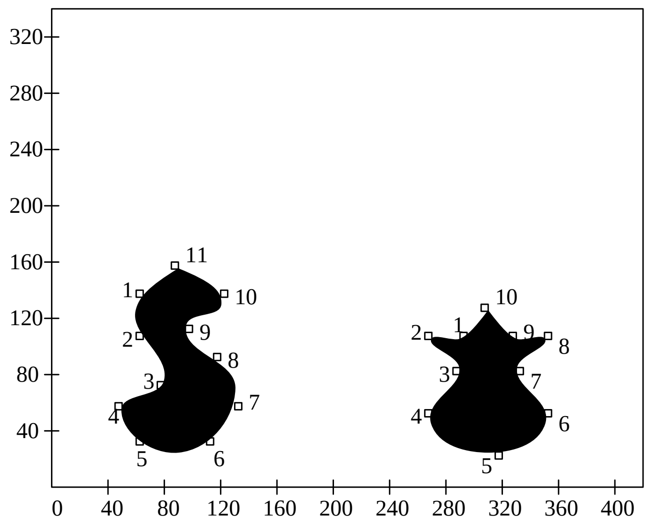


Figure 16: Two reference objects $C5$ and $C6$.

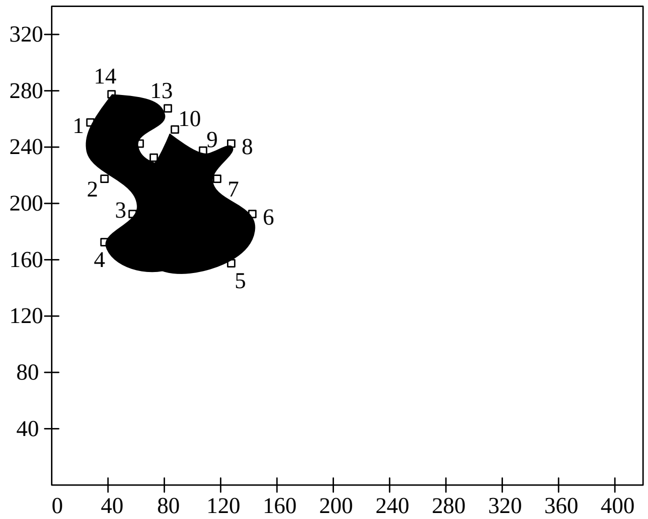


Figure 17: Test object $C7$ consists of $C5$ and $C6$.

Example 8:

Figure 16 shows two reference objects $C5$ and $C6$ and Figure 17 shows the test object $C7$ obtained by overlapping $C5$ (translated by $(-22, 126)$ and rotated by 20^o) and $C6$ (translated by $(-213, 125)$ and rotated by 15^o). Note that a significant amount of features of $C5$ and

$C6$ disappear in this case. The dominant points for $C5$, $C6$ and $C7$ are indexed on the contours shown in Figures 16 and 17. The $\{(\theta, p)\}$ and $\{\delta\theta\}$ for $C5$, $C6$ and $C7$ are shown in Figure 18.

	object C5			object C6			object C7		
	P	θ	δθ	P	θ	δθ	$\hat{P}$	$\hat{\theta}$	$\delta\hat{\theta}$
1	65	6	7	127	85	45	29	25	10
2	88	13	98	274	40	100	196	35	98
3	40	111	93	-172	140	78	67	133	43
4	59	18	73	160	62	51	148	90	40
5	24	91	51	-98	113	89	-33	130	89
6	–65	142	142	341	24	108	233	41	105
7	121	0	42	-156	132	45	26	146	46
8	139	42	66	119	87	45	221	100	44
9	85	108	64	313	42	93	253	56	97
10	177	44	83	-135	135	50	24	153	122
11	72	127	121				176	31	96
12							165	127	64
13							274	63	83
14							115	146	121

Figure 18: Example 8: $\{(\theta, p)\}$ values of C5, C6 and C7

It should be noted that due to the discretization of the image and quantization errors, $\delta\hat{\theta}_j = \delta\theta_i \pm \Delta\Psi$ or $\delta\theta_i + \delta\hat{\theta}_j = 180^o \pm \Delta\Psi$, where $\Delta\Psi \leq 2\Delta\tilde{\phi}$, and $\Delta\tilde{\phi}$ is the deviation angle (see Section 4.4). In this example, $\Delta\tilde{\phi} = 3^o$. The results of comparing $\{\delta\theta\}_{C5}$ and $\{\delta\theta\}_{C6}$ with $\{\delta\hat{\theta}\}_{C7}$ are stored in $LIST_{C5}(12)$, and $LIST_{C6}(5)$, where

$$LIST_{C5}(12) = \{\ 64,\ 83,\ 121,\ 10,\ 98\ \},$$
$$LIST_{C6}(5) = \{\ 89,\ 105,\ 46,\ 44,\ 97\}.$$

Since these two sets are disjoint, we conclude that the dominant points of the test object $C7$ indexed as 1, 2, 12, 13, and 14 belong to object $C5$, and the dominant points indexed as 5, 6, 7, 8, and 9 belong to object $C6$.

Since the correspondence between one candidate object and the test object has been determined in the above procedure, the rotation angle can be obtained using the average of the differences between θs and corresponding $\hat{\theta}$s. Hence, the rotation angle ϕ for $C5$ is 19.5^o and for $C6$ is 15^o. We follow the same procedure of estimating the translation vector as in Section 4.2. The computed results of translation terms, intrinsic terms, and $\Delta p(\theta)$s for both $C5$ and $C6$ are listed in Figure 19. The translation vector is computed by voting in the $x-y$ plane in an area of 60×60 pixels around the centroid of the polygonal approximation of $C7$. The estimated (x^*, y^*) for $C5$ is $(64, 218.5)$ and for $C6$ is $(101, 196)$. Therefore, the translation vectors for $C5$ and $C6$ are $(-26, 130.5)$ and $(-210, 126)$, respectively. Please refer to the summary of the motion estimation results in Figure 14.

Object C5				Object C6			
index	translation term T	p_0	Δ p	index	translation term T	p_0	Δ p
1	98.7	-33.7	62.7	5	-57	-41	8
2	107.5	-19.5	215.5	6	313	28	205
9	55.74	29.26	135.74	7	-156.1	0.1	25.9
10	125.8	51.2	223	8	86.32	32.7	188.3
11	16	56	59	9	278.2	34.8	218.2

Figure 19: Example 8: $T(\theta)$, $p_0(\theta)$ and $\Delta p(\theta)$ values for C5 and C6

6 Computational Complexity

In this section, we analyze the overall complexity of the proposed motion estimation technique. We consider the computational complexity of three parts, namely, the preprocessing work, computation of the straight line approximation of the boundary, and the estimation of motion parameters. Assume that the input image is of size $N \times N$, the length of the object contour is P pixels, and the number of dominant points is n.

The preprocessing work includes smoothing the boundary, tracing the contour and detecting dominant points. Since we directly work on the raster input pixels to smooth the boundary, the computational complexity is $O(N^2)$. Searching the region of support for each dominant point, and computing the curvature for each edge pixel on the contour have a complexity of $O(P)$. The time to detect dominant points within each region of support is $O(P)$. Therefore, the preprocessing work has a complexity of $\approx O(N^2)$ since $N^2 >> P$.

The straight line approximation of the boundary is determined by computing the approximate $(\tilde{\theta}, \tilde{p})$ of the line connecting adjacent dominant points and refining the values using SLHT. The time to compute the $(\tilde{\theta}, \tilde{p})$s is $O(n)$, and $O(n) < O(P)$. Let $\hat{m_\theta}$ denote the number of quantization levels for the θ values during the computation of SLHT. Since the SLHT of the pixels between two adjacent dominant points is computed around the initially computed $\tilde{\theta}$ values (see Section 3.3), $\hat{m_\theta}$ is a small value (usually less than 10), and the complexity of this step is $O(\hat{m_\theta} * P)$. Hence, the straight line approximation of the boundary has a complexity of $O(n + \hat{m_\theta} * P) \approx O(\hat{m_\theta} * P)$.

In motion estimation, the time required to estimate the rotation angle is $O(n^2)$, and the time required to compute the intrinsic terms and translation terms is $O(n)$. Finally, the voting algorithm to determine the displacement vector has a complexity of $O(n + b)$, where b is the time to locate the peak in the $x - y$ plane, and $b << N$ (see Section 4.3). Therefore, the total motion estimation cost is $O(n^2 + n + n + b) \approx O(n^2 + b)$.

For comparison purposes, we consider the matching algorithm proposed by Pao, Li and Jaykumar.[(12)] In their work, the overhead includes computing the SLHT and filtering the $\theta - p$ space. Since the SLHT on all the pixels is computed in the range of θ values from 0^o to 180^o, the number of quantization levels denoted as m_θ is equal to 180 for $\Delta\theta = 1^o$ (quantization interval). Note that m_θ (=180) is significantly larger than $\hat{m_\theta}$ ($\approx$ 10) of our method. The complexity of computing SLHT is $O(m_\theta * P)$, and the total overhead is approximately $O(N^2 + m_\theta * P)$. The total estimation cost is approximately $O(m_\theta^2 * K_D + N^2)$, where K_D is the average number of points in each column of the reference signature.[(12)] We summarize the results of comparison in Figure 20.

Since $m_\theta >> \hat{m_\theta}$, the total overhead of Pao et al's method, $O(N^2 + m_\theta * P)$, is larger than the overhead due to preprocessing and straight line approximation of the boundary in our method. The estimation cost of Pao et al's method, $O(m_\theta^2 * K_D + N^2)$ is also larger than that of our method, $O(n^2 + b)$ since $N >> n$ and $m_\theta^2 * K_D > b$. Thus our method involves much less computation than Pao et al's method. This matches the observation that

	preprocessing work	computing SLHT	estimation cost
proposed method	$O(N^2)$	$O(\hat{m}_\theta * P)$	$O(n^2 + m^2)$
Pao et al's method	$O(N^2 + m_\theta * P)$		$O(m_\theta^2 * K_D + N^2)$

Figure 20: Comparison of the computational complexities of Pao et al's method and our method for motion estimation of nonoverlapping objects.

a compact representation results in fewer computations provided that the preprocessing is not excessive.

When objects overlap, the computational complexity of our scheme can be analyzed similarly. Let there be $k-1$ candidate objects and one new (test) object. Let P_d represent the average number of contour pixels in an object and n_d the average number of dominant points in an object. The preprocessing time complexity is $O(N^2 + k * P_d + k * P_d) \approx O(N^2)$ since $N^2 >> k * P_d$. The total computational time for straight line approximation of the boundary is $O(k * n_d + k * \hat{m}_\theta * P_d) \approx O(k * \hat{m}_\theta * P_d)$.

Motion estimation of occluded objects invloves recognizing each part involved in the occlusion, and determining the motion parameters of each of the candidate objects. The complexity of computing the $\{\delta\theta\}$s of a candidate object is $O(n_d)$, and that of matching the $\{\delta\hat{\theta}\}$ of the new object with the $\{\delta\theta\}$ of a candidate object is $O(n_d^2)$. The procedure to estimate the rotation angle of a candidate object is $O(n_d)$, since the correspondence between the angles of a candidate object and the test object has already been determined. The complexity of computing the intrinsic terms and translation terms per object is $O(n_d)$, and that of the voting algorithm to determine the displacement vector is $O(n_d + b)$. Therefore, the estimation cost per candidate object is $O(n_d + n_d^2 + n_d + n_d + n_d + b) \approx O(n_d^2 + b)$, and the total cost is $O(k(n_d^2 + b))$. We summarize the complexity results of the proposed method and Pao et al's method in Figure 21.

A comparison of the two methods shows that the computational complexity of our method

	preprocessing work	computing SLHT	estimation cost
proposed method	$O(N^2)$	$O(k * \hat{m}_\theta * P_d)$	$O(k * n_d^2 + k * m^2)$
Pao et al's method	$O(N^2 + k * m_\theta * P_d)$		$O(k * m_\theta^2 * K_D + k * N^2)$

Figure 21: Comparison of the computational complexities of Pao et al's method and our method for motion estimation of overlapping objects.

is lower than that of Pao et al's method for the problem of overlapping objects. The reduction in the complexity is significant when the number of candidate objects is large.

7 Conclusions

In this paper, we have proposed a novel feature-domain motion estimation technique based on the straight-line Hough Transform (SLHT). The proposed method is based on compact representation of the object and requires significantly fewer computations compared to Casasent et al's method(11) and Pao etal's method.(12) Specifically, the representations in reference (11) and (12) consist of tangent lines on every boundary pixel, while our representation consists of a small set of straight lines with each line approximating the boundary pixels between two dominant points. We reduce the overhead of detecting dominant points by proposing a new computationally-efficient method to detect these points in a single pass during contour tracing. Moreover, our method does not require any thresholding during the computation of SLHT, since each group of pixels gives rise to a single peak (corresponding to a single line). The compact object representation also enables us to efficiently determine the translation and rotation parameters, because of the reduction in the size of the parameter space. An extension of this method to estimate the rotation and translation parameters for overlapping objects has also been presented. The proposed motion estimation scheme is currently being mapped onto hardware using Mentor Graphics tools.

References

[1] J.Limb and J.Murphy, "Estimating the velocity of moving images in TV signals," *Comput. Graph. Image Proc. 4*, pp. 311-327, 1975.

[2] C.Cafforio and F.Rocca, "Tracking moving objects in TV images," *Signal Proc. 1*, pp. 133-140, 1979.

[3] T.S.Huang (ed.), "Image Sequence Analysis." Springer-Verlag, New York, 1981.

[4] T.S.Huang (ed.), "Image Sequence Processing and Dynamic Scene Anlysis." Springer-Verlag, New York, 1983.

[5] G.R.Legters, JR. and T.Y.Young, "A mathematical model for computer image tracing", *IEEE Transactions on PAMI*, vol. 4, no. 6, pp. 583-594, 1982.

[6] R.P.Futrelle and G.C.Speckert, "Extraction of motion data by interactive image processing," in *Proc. IEEE Conf. Pattern Recognition and Image Processing*, pp. 405-408 May 1978.

[7] T.Y.Young and K.S.Fu, "Handbook of Pattern Recognition and Image Processing", Academic Press, Orlando, FL, 1986

[8] T.Y.Young, "Handbook of Pattern Recognition and Image Processing: Computer Vision", Academic Press, Orlando, FL, 1994

[9] R.O.Duda and P.E.Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Comm. ACM,* vol. 15, no. 1, pp. 11-15 1972.

[10] S.Tsuji and F.Matsumoto, "Detection of ellipses by a modified Hough Transform," *IEEE Trans. Comput.,* vol. 27, pp. 777-781, 1978.

[11] D.Casasent and R.Krishnapuram, "Curved object location by Hough transformations and inversions," *Patt. Recogn.,* vol. 20, no. 2, pp. 181-188, 1987.

[12] D.C.W.Pao, H.F.Li, and R.Jayakumar, "Shapes recognition using the straight line Hough transform: theory and generalization," *IEEE Trans. on PAMI*, vol. 14, no. 11, pp. 1076-1089, Nov. 1992.

[13] D. Langridge, "On the compuation of shape," in *Frontiers of Pattern Recognition*, S. Watanabe, Ed. New Yorkk: Academic, pp. 347-365, 1972.

[14] C.H.Teh and R.T.Chin, "On the detection of dominant points on digital curves," *IEEE Transactions on PAMI*, vol. 11, no. 8, pp. 859-872, 1989.

[15] H.C.Liu and M.D.Srinath, "Corner detection from chain code," *Patt. Recogn.,* vol. 23, no. 1/2, pp. 51-68, 1990.

[16] N.Ansari and K.W.Huang, "Non-parametric dominant point detection," *Patt. Recogn.,* vol. 24, no. 9, pp. 808-813, Aug., 1994.

[17] N.Ansari and E.J.Delp, "On detecting dominant points," *Patt. Recogn.,* vol. 24, no. 5, pp. 441-451, 1991.

[18] C.Fermuller and W.Kropatsch, "A syntactic approach to scale-space-based corner description," *IEEE Transactions on PAMI*, vol. 16, no. 7, pp. 748-751, July, 1994.

[19] S.C.Pei and C.N.Lin, "The detection of dominant points on digital curves by scale-space filtering," *Patt. Recogn.,* vol. 25, no. 11, pp. 1307-1314, 1992.

[20] A.Pikaz and I.Dinstein, "Using simple decomposition for smoothing and feature point detection of noisy digital curves," *IEEE Transactions on PAMI*, vol. 16, no. 8, pp. 430-449, 1992.

[21] F.Mokhtarian and A.Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Transactions on PAMI*, vol. 8, no. 1, pp. 34-43, 1986.

[22] A.Rattarangsi and R.T.Chin, "Scale-based detection of corners of planar curves," *IEEE Transactions on PAMI*, vol. 14, no. 4, pp. 430-449, 1992.

[23] M.H.Han and D.Jang, "The use of maximum curvature points for the recognition of partially occluded objects," *Patt. Recogn.,* vol. 23, no. 1/2, pp. 21-33, 1990.

[24] M.H.Han, D.Jang and J.Foster, "Inspection of 2-D objects using pattern matching method," *Patt. Recogn.,* vol. 22, no. 5, pp. 567-575, 1989.