Rectangular Object Tracking
Based on Standard Hough Transform

Thuy Tuong Nguyen, Xuan Dai Pham and Jae Wook Jeon, Member, IEEE
School of Information and Communication Engineering
Sungkyunkwan University, Suwon, Korea
ntthuy@skku.edu, pdxuan@ece.skku.ac.kr, jwjeon@yurim.skku.ac.kr

Abstract—Object tracking in computer vision is important. Numerous research papers have been published about this problem. Few references relating to tracking shaped objects via the Hough Transforms exist. This paper provides a method based on the Standard Hough Transform to track rectangular objects. Our method works on edge images obtained by applying the Canny edge detector to the source image. The rectangular object in the edge image is then extracted based on the Standard Hough Transform and related conditions of the rectangle. Afterwards, we shift the region of interest to the new position of the tracked object and continue to detect the rectangle in successive frames. Our technique can be run in real-time due to its low computation time and simplicity.

Index Terms—Object tracking, rectangular object tracking, Standard Hough Transform.

I. INTRODUCTION

Tracking objects and image features is an important task that can be applied to many research areas. Tracking a moving object involves the detection and localization of this object in an image sequence. It can be categorized into one of two primary groups. The first group relates to the tracking of known objects. In this category, the object to be tracked has to be allocated in advance. The second group involves tracking unknown moving objects. This type of tracking is commonly referred to as motion-based tracking. In this paper, a method of tracking a known object through its features is introduced.

Many publications are related to known object tracking. Two of the most robust tracking techniques are the particle filter technique [1], [2], [3], [4] and the mean shift technique [5], [6], [7], [8]. The research in [1] and [2] is based on the principle of color histogram distance, within a probabilistic framework. The authors use a particle filter to improve handling color clutter in the background. Yang [3] introduces an efficient and robust visual object tracking algorithm based on the particle filter. The object is tracked based on the combination of the contour-based model and color-based model. The authors reduce the computation time of the particle filter using the integral image. An advance of the particle filter is presented in [4]. This paper proposes a method to associate dynamic shape information to the particle filter used to track deformable objects. In [5] and [6], Comaniciu makes use of a metric obtained from the Bhattacharyya coefficient, as a similarity measure, and uses the mean shift procedure for optimization. An extension of the mean shift algorithm is presented in [7]. Instead of only considering the position of the tracked object, this extension also estimates the scale that adapts to the shape of the tracked object. The work in [8] extends the traditional mean shift algorithm, presenting the use of asymmetric kernels. This research contributes to the online estimation of the kernel scale and orientation at each mean shift iteration.

Little known work relates to tracking objects via the Hough Transforms [9], [10], [11]. The Dynamic Velocity Hough Transform (DVHT), to track parametric objects, is proposed in [9]. This algorithm extends the Velocity Hough Transform to cope with arbitrary motion. The authors provide a cost function and maximize it to follow the object. Three constraints relate to the cost function: the trajectory has to pass through the points of the Hough space with the maximum structure evidence, the direction between two consecutive frames must be smooth, and the points in the Hough space corresponding to large changes in velocity are penalized. The comparison between DVHT and Standard Hough Transform [12], [13] to track a ball in a table-tennis image sequence is also provided. The result from DVHT is superior.

Conversely, [10] and [11] utilize the standard Hough transform algorithm that detects straight lines. The joint probability density function, used in [10], represents the probability of the existence of a pair of parallel lines. It is a product of three probabilities: two probabilities of the existence of individual lines, and the probability that the set of parameters form a parallel pair. In [11], French combines the condensation tracking algorithm, also known as the particle filter algorithm, with the Hough transform to create a tracking technique that is efficient and robust under illumination changes, occlusion and distractions.

In this paper, a surprisingly simple yet powerful method, based on the Standard Hough Transform, is proposed to track rectangular objects. It works on the edge image obtained by applying the Canny edge detector to the source image. The rectangular object in the edge image is then extracted based on the Standard Hough Transform and related conditions of a rectangle. Finally, the region of interest is shifted to the
new position where the tracked object is located.

This paper is organized as follows. The next section details
our proposed method. Section III presents the discussion. Experimental results are in Section IV. This paper is drawn
to a conclusion and future work presented in Section V.

II. THE PROPOSED METHOD

Our method tracks an object based on its boundary. This type of tracking technique usually utilizes edges as the representative feature [14]. An important characteristic of edges is that they are less sensitive to illumination changes, when compared to color features. Edge detection is the first step of our proposed method. The Canny edge detector is selected due to its simplicity and accuracy. Then, the search window is defined manually at the position of the object to be tracked. We attempt to define a window \( W \) outside the tracking object, or tracking rectangle, \( O \) where the center of the window coincides with the center of the tracking rectangle. The related equations are shown as follows:

\[
\begin{align*}
O &= (x_o, y_o, w_o, h_o), \\
R &= (x_r, y_r, w_r, h_r), \\
W &= (x_w, y_w, w_w, h_w) = (x_r, y_r, w_r, h_r, r_w), \\
(x_r, y_r) &= (x_o, y_o), \\
(x_w, y_w) &= (x_r, y_r), \\
(x_o, y_o) &= \left( \frac{\max(X) + \min(X)}{2}, \frac{\max(Y) + \min(Y)}{2} \right), \\
w_r &= \max(X) - \min(X), \\
h_r &= \max(Y) - \min(Y), \\
w_w &= w_r + 2r_w, \\
h_w &= h_r + 2r_w,
\end{align*}
\]

where \( x_i, y_i, w_i, h_i \) are the center, width and height of \( i \); \( R \) is the "closest" rectangle of the object to be tracked (Fig. 1); \( X = \{x_0, x_1, ..., x_{N-1}\} \) and \( Y = \{y_0, y_1, ..., y_{N-1}\} \) are two sets of \( x \) and \( y \) coordinates of the edge image of tracking rectangle, \( N \) is the total number of pixels; \( r_w \) is the distance between the two borders of \( R \) and \( W \). The distance \( r_w \) depends on the velocity of the moving object. The center \( (x_o, y_o) \) and vertices of the object are the information propagated between the frames to support the tracking process. The method to calculate the vertices will be presented in step 3 of section B.

In the next sections, the Standard Hough Transform algorithm is applied to the search window \( W \) to detect straight lines, and the related conditions of a rectangle help detect the rectangular object in the corresponding region. Afterwards, the object to be tracked is determined and the stage of defining the search window is revisited.

A. Straight line detection using Standard Hough Transform

The Hough Transform [12], [13] has been the most popular algorithm utilized to extract global features, such as straight lines, circles, and ellipses, from an image. This algorithm uses an array called parameter space to detect the existence of a line. For each pixel in the edge image, the Hough transform algorithm determines if there is sufficient evidence of a line at that pixel. It calculates the parameter values and accumulates the cells in the parameter space according to all the pixels. Then, by finding the cells with the highest values, the most likely lines can be extracted. The simplest way to find these peaks is to apply some form of threshold. Finally, the segments are extracted using the information obtained from the maximum positions.

The basic Hough Transform, also called the Standard Hough Transform (SHT) [12], [13], has established itself as the default technique for straight line Hough transform evaluation. Its popularity comes from its robustness to noise and the very simple algorithmic implementation of SHT. In SHT all the points \( (x, y) \) are calculated using the equation expressed in polar coordinates

\[
\rho = x \cos \theta + y \sin \theta,
\]

to obtain \( \rho \) with a change in the succession of \( \theta \) in the parameter space. With the values \( (\rho, \theta) \) determined, the SHT algorithm then looks for the accumulator’s cells into which
the parameters fall, and increases the values of those cells. The transformation of an image \( I \) to a Hough space \( H \) can be represented as

\[ I(x, y) \rightarrow H(\rho, \theta). \]

After some edge pixels considered to constitute a straight line have been accumulated, the distribution of the parameter space is shaped like a butterfly [15]. The most likely straight line in the image domain is represented by the highest value in the parameter space. The peak lies at the center of the butterfly. When all the accumulated points form a chaotic parameter space, several overlapping butterflies are obtained. The present accumulator has peaks that may have values greater than the specified threshold. The simplest way to find these peaks is to compare the values of the peaks with the threshold value. Then the peaks with the highest values are compared to other peaks in the neighborhood using predefined windows. Straight lines are detected via these peaks:

\[ L_{\rho\theta} = \{ (\rho, \theta) | C[(\rho, \theta)] > T_{\text{vote}}, i = 0 \ldots M - 1 \}, \]

or

\[ L_{\rho\theta} = \{ l_i | C[l_i] > T_{\text{vote}}, i = 0 \ldots M - 1 \}, \]

where \( L_{\rho\theta} \) is a set of detected straight lines that satisfy the condition of threshold value; \( C[(\rho, \theta)] \) is the accumulating value at the cell \((\rho, \theta)_i\); \( T_{\text{vote}} \) is the threshold value; and \( M \) is the number of detected lines. The vote threshold \( T_{\text{vote}} \) is specified so the line, whose length is greater than one fourth of the diagonal of searching window, is detected:

\[ T_{\text{vote}} = \frac{\sqrt{w^2 + h^2}}{4}. \]

For simplicity, we define an inner circle \( C_{\text{inner}} = (x_{ci}, y_{ci}, r_{ci}) \) for the search window \( W \). The inner circle is an attempt to eliminate superfluous pixels that are inside the tracked object. This work reduces the computation time of the Hough Transform; it reduces noise that interferes with the accuracy of the tracking process. The center of this circle also coincides with the search window,

\[ (x_{ci}, y_{ci}) \equiv (x_w, y_w), \]

and the radius \( r_{ci} \) is calculated as

\[ r_{ci} = \frac{\sqrt{w^2 + h^2}}{10}. \]

Fig. 2 shows an illustration of the inner circle. We see that this circle is useful when there is noise inside the object (Fig. 2(b)). Fig. 2(c) is shown with the elimination of noise at the center of the window. In Fig. 2(a), whether or not the inner circle is applied, the result is also satisfactory.

B. Rectangular object detection based on detected straight lines

This section contains four main steps to detect the rectangular object: finding parallel pairs of lines, eliminating disturbed lines, finding lines for rectangles, and eliminating disturbed lines that belong to rectangles.

- **Step 1—Finding parallel pairs of lines.**
  The straight lines detected from the previous section are scanned. All the parallel pairs of lines from the detected line list are stored in \( P_L \). The lines that are isolated, or not parallel to any lines, are eliminated. The result at this step can be expressed as follows:

\[ P_L = \{ (l_i, l_j) | \forall l_i, l_j \in L_{\rho\theta}, |\theta_i - \theta_j| \leq T_{\theta} \}, \]

where \( T_{\theta} \) is a threshold of \( \theta \).

- **Step 2—Eliminating disturbed lines.**
  An assumption is provided in this step: the number of noises in the search window is adequate and thus several straight lines are detected. The list of lines detected by SHT is sorted in descending order with respect to accumulating value. This work leads to an assumption that the most likely line has the biggest accumulating value. Consequently, we select the first line \( l_i \) in the list of detected lines, as a criterion to eliminate disturbed lines. From this criterion, we attempt to find the first line \( l_j \) that is perpendicular to \( l_i \) in \( P_L \). The two \( l_i \) and \( l_j \) are used to delete the pairs that contain lines parallel to these two standard lines. Finally, we have \( P'_L \) that contains the best candidates to form rectangles.

- **Step 3—Finding lines for rectangles.**
  All lines from \( P'_L \) are grafted with respect to the perpendicular condition:

\[ |\theta_i - \theta_j| \in \left[ \frac{\pi}{2} - T_{\theta}, \frac{\pi}{2} + T_{\theta} \right], \]

where \( i \) and \( j \) are the indices of pairs in \( P'_L \); \( a \) is 0 or 1 corresponding to the first and the second line in each pair.

Having grafted the pairs, we can compute their intersection points. A vertex is obtained from each pair of two
perpendicular lines. Assigning the index of one line as $i$, and the index of its perpendicular line as $j$, we have:

$$
\begin{align*}
\rho_i &= x \cos \theta_i + y \sin \theta_i \\
\rho_j &= x \cos \theta_j + y \sin \theta_j \\
\Rightarrow \quad x &= \frac{\rho_j \sin \theta_i - \rho_i \sin \theta_j}{\cos \theta_i \sin \theta_j - \cos \theta_j \sin \theta_i}, \\
y &= \frac{\rho_i - x \cos \theta_i}{\sin \theta_i},
\end{align*}
$$

where $(x,y)$ is the vertex position.

• **Step 4—Eliminating disturbed lines that belong to rectangles.**

In the previous step, the calculated intersection points may be outside the search window. It is necessary to eliminate the rectangle that contains at least one vertex positioned outside the search window. Fig. 3 shows four cases of rectangle to be eliminated: four vertices are outside the window, three vertices are outside, two vertices are outside, and one is outside.

As a consequence the method used to find parallel pairs of lines in Step 1 and to find lines for rectangles in Step 3, each foursome of successive lines belongs to one rectangle (Fig. 4). Therefore, a rectangle—four lines—is deleted if at least one vertex belonging to it is outside.

### III. Experimental Results

Experiments are conducted on both synthetic and real image sequence. The synthetic sequence is with a moving rectangle immersed in a noisy background. The real sequence captured from a camera has one moving box, of which one side will be tracked. The system was implemented on a 3 GHz Pentium IV PC with 1 GB RAM, using the Windows XP SP2 operating system. All images used in our experiment are $320 \times 240$ pixels.

The first experiment is run on a synthetic sequence of 99 frames. The background containing pixel-noise in each frame is generated using a Gaussian Random Number Generator (RNG). The mean, variance, and number of noise pixels of the RNG are 0, 200, and 8000, respectively. The moving object is a rectangle initially placed along the bottom-left to top-right diagonal of the image. The rectangle is 53 pixels in width, and 39 pixels in height. There is no noise inside this object. Fig. 5 shows four sample frames from the sequence of 99 frames. The first frame of the sequence is shown in Fig. 5(a). The object moves with an approximate angle of 30 degrees to the left. Frame 37 shows the object before it touches the right side. Then, the object moves to the left with an approximate angle of 45 degrees. It hits the top side at frame 51. Finally, the rectangular object attempts move back along the initial direction. In frame 93, the object is next to the left side. Our method is used to track this rectangular object. The corresponding parameters of the search window are $(x_w, y_w) = (80, 170)$ and $r_w = 14$ pixels. In SHT, the parameters to detect lines are $\Delta \rho = 1$ pixel, $\Delta \theta = 0.01$ radian, and number of maximum detected lines $\text{lines}_{\text{max}} = 2101$.  

![Fig. 3. Four cases of rectangle to be eliminated. (a) Four, (b) three and (c) two vertices are outside the search window; (d) one vertex is outside.](image)

![Fig. 4. List of detected lines in which each of four successive lines is grouped to form a rectangle.](image)

![Fig. 5. Four sample frames of the synthetic sequence. (a) Frame 1, (b) Frame 37, (c) Frame 51, and (d) Frame 93.](image)
6 lines. $T_\theta = 0.1$ radian in the section of rectangle detection. The result demonstrates the proposed method perfectly tracks the moving object throughout the sequence of frames. Four detected lines representing a rectangle fit to four edges, successfully tracking the object. The result of four sample frames is shown in Fig. 6, where the border of the detected rectangle is a red color. The frame rate of the resultant sequence is approximately 13 fps. In similar experiments, when we change the number of noise pixels to the values that are greater than 8000, the tracking rectangle will be missed.

The second experiment is run on a real sequence of 61 frames. These frames were captured using a Logitech 961403-0311 QuickCam Fusion camera. The frame sequence shows a man holding a box. The man moves the box from the center of the view, to the right and then returns to the initial position. One side of the box is tracked. It has the shape of a rectangle 95 pixels in width, and 125 pixels in height. There is no noise inside this rectangle. The left column of Fig. 7 shows four sample frames from the sequence of 61 frames. The first frame where the object was placed initially near the center of the view is shown in Fig. 7(a). Frames 20, 40 and 60 show the next positions of the object. The proposed method tracks the rectangle that is one side of the box. The parameters used in this experiment are the same as in the previous one except $(x_w, y_w) = (134, 121)$ and $r_w = 15$ pixels. The result of four sample frames in the left column is shown in the right column of Fig. 7. The rectangle that is tracked correctly is bordered in red. If the rectangle is missed, a pink border is drawn at the previous position. Fig. 7(d) shows that the object is missed and is bordered by a pink rectangle. 45 of 61 (74%) tracked frames are detected. The frame rate of the resultant sequence approximates 11 fps.

Although the 74% detection rate is not very convincing with chosen parameters, but we need to consider that noise exists in the real environment. The noise can come from illumination changes or background movement. Noise affects the result of edge detection that causes rectangle detection in our proposed method. Therefore, it is not easy to achieve 100% detection.

### IV. Discussion

Our proposed method can track the object successfully in both synthetic and real frame sequences. Three problems need to be analyzed. The first problem relates to the velocity of the moving object. The second concerns background noise. The third problem concerns noise inside the object.

A search window is defined in our method. It is used to limit the search space for the Hough Transform algorithm and to eliminate a great deal of noise outside the object. It reduces computation time and increases the accuracy of the method. The width and height of this window are adapted, depending on the size of the tracked object. The width and height also depend on the velocity of the object. If the object moves fast, we need to set this size larger. Conversely, we can lessen this size for slow moving objects.

When the object moves, it may meet a background that has a similar intensity or color. In this case, the edge detector might create noise in the intersection region between the object and the background. Fig. 7(c) and Fig. 7(d) show the original image and its tracked result. We see in the right hand side, the edges of the tracked object and the temperature regulator are mistaken.

The third problem relates to noise inside the tracked object. We sketch this problem in Fig. 8. Fig. 8(a) shows a rectangle with a word inside. This word is considered as noise. Fig. 8(b) shows the result of our method. Two rectangles are detected in this case. The red and solid border represents the correctly detected object, and the black and dashed border represents the noise-rectangle. The center of this object will not be at the center of the correctly detected rectangle. Fig. 8(c) and Fig. 8(d) display a green circle as the center of the object. The size of the search window is calculated based on the object’s center. In this case, the window may miss one or more edges of the tracked rectangle. We will lose the object to be tracked in the presence of this type of noise.

### V. Conclusion and Future Work

This paper proposed a method based on the Standard Hough Transform to track rectangular objects. Our method can track the object successfully in both synthetic and real frame sequences. We also discussed some related problems. Our technique can be run in real-time, because of its low computation time and simplicity.

For future work, we plan to overcome the method’s shortcomings as discussed. We will attempt to extend the idea to track arbitrarily shaped objects.
Fig. 7. Four sample frames and four resultant frames of the real sequence. (a) Frame 1 and (b) its result, (c) Frame 20 and (d) its result, (e) Frame 40 and (f) its result, (g) Frame 60 and (h) its result.

Fig. 8. An example of noise inside the object. (a) The object to be tracked, (b) the result of our method in tracking the object, (c) the correct result and its center, and (d) the incorrect result and its center.

ACKNOWLEDGMENT

This research was performed as part of the Intelligent Robotics Development Program, one of the 21st Century Frontier Research and Development Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

REFERENCES