Implementing Rectangle Detection using Windowed Hough Transform

Akhil Singh, Music Engineering, University of Miami

Abstract—This paper implements Jung and Schramm’s method to use Hough Transform for rectangle recognition using a few pre-processing methods and performing a windowed application so that the algorithm can perform faster. Images are scanned with a sliding Hough transform window, peaks of the transform are extracted (which correspond to line segments) and rectangles are detected based on a certain geometric condition. The algorithm proposed was effective in identifying rectangles in synthetic images and a few ‘not so complex’ natural images. This method fails in the case of overlapped images. An alternate method was used for such images.

Index Terms—Hough Transform, Rectangle Detection, Image Processing

1 INTRODUCTION

Object detection in general has many applications like detecting number plates of vehicles, in microbiology (to detect the number/type of cells), aerial imaging, etc. In this project, we particularly talk about detection of rectangles in a given image. Implementation and results of windowed Hough Transform suggested by Jung and Schramm are shown and also a few modifications to the original algorithm are suggested and implemented.

The easiest way to detect rectangles is to tackle it from the basic level of lines and edges. Such a method will be limited to parallel and orthogonal lines. It would fail miserably in the cases where the images are angled, do not have perfect sharp edges/intersections, etc. .

There have been different methods to tackle this problem. Windowed Hough transform[2] was found to be a very efficient way to detect rectangles. The paper clearly mentions that false detection results when aligned rectangle are close to each other. In this project, we start with the implementation of their idea and modifying it to get our desired results.

2 IMPLEMENTATION

2.1 Hough Transform

Hough transform is a feature extraction technique used in image analysis and digital image processing to find geometrical shapes in an image by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform[1]. The idea of the transform is the fact that any line on an xy plane can be described as \( \rho = x \cos \theta + y \sin \theta \) where \( \rho \) is the normal distance and \( \theta \) is the normal angle of a straight line, making the line a 2D function \( C(\rho, \theta) \) that represents the number of edge points satisfying the equation. Fig. 1 shows the graphical representation of the transformation.

Fig. 1: Representation of a straight line in Hough Parameters[1]
The local maxima of the 2 dimensional line equation obtained after the transform can be used to detect the straight line segments passing through edge points. These edge points and line intersections help us in determining various geometrical structures based on the generic equations. In our case the geometrical structure being a rectangle.

2.2 Rectangle Patterns in the Hough Space

In this section, we discuss the implementation that Jung and Schramm suggested to detect rectangles in Hough space. All the images from the following section are taken from their paper. The generic rectangle has 5 degrees of freedom: two coordinates of the center, width, height and orientation. Having a 5D accumulator array to compute would be computationally very expensive and time taking. The paper talks about an easier implementation which is computational cheaper but efficient.

Consider a rectangle with vertices \( P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = (x_3, y_3), \) and \( P_4 = (x_4, y_4) \) with \( P_1P_2, P_3P_4 \) being parallel sides with length \( a \), as well as \( P_2P_3 \) and \( P_4P_1 \) with length \( b \). Also assume that the coordinate system of is located in the center of the rectangle, as shown in Fig.2.

The above mentioned relations are valid only if there is just one rectangle present in the region. Multiple edges or rectangles wouldn’t give accurate results because presence of any other structure in the image would have a global influence in the Hough space. The relations 1, 2 and 3 are more generic and should work globally.
2.3 Jung and Schramm’s algorithm

The basic idea of the algorithm proposed by them, was to apply Hough transform on the image, identify peaks which satisfy the conditions mentioned in the previous section and detect the rectangle. As mentioned earlier, the conditions fail when there are additional edges in the area. To overcome this defect, they used a windowed approach.

2.3.1 Windowed Hough Transform

Consider a rectangle centered at \((x_0, y_0)\). For the algorithm to be able to detect the rectangle, the windowed region must be large enough to contain all the edges of any possible rectangle centered at the given location. On the other hand, it should be as small as possible to avoid edges belonging to other structures.

To tackle this problem, they proposed a suitable search region to be a ring with internal diameter \(D_{\text{min}}\) and external diameter \(D_{\text{max}}\). The choice of these parameters is user defined where \(D_{\text{min}}\) should be equal to the smallest possible side of the rectangle and \(D_{\text{max}}\) should be approximately equal to the largest possible diameter. They claim that such a choice of parameters will ensure that any rectangle in the image will have all of its edges within the search region. The windowed Hough transform implementation is shown in Fig. 4.

Fig. 4 shows the edge map of synthetic image computed with Canny’s operator and also the choice of parameters for the ring shaped search region with internal and external diameters to be \(D_{\text{min}}\) and \(D_{\text{max}}\).

This approach to the problem is very efficient in the cases of rectangles which are spaced out. In the cases where rectangles of different sizes are very close to each other, such an approach would give us multiple edges. Later in the report, an alternate way to improve this situation is suggested and implemented.

Once the \(D_{\text{min}}\) and \(D_{\text{max}}\) are specified, Hough transform is computed using quantized orientations of \(\theta\) and \(\rho\). Discretization steps \(d_\theta\) and \(d_\rho\) were calculated based on the size of the image as suggested by Furukawa and Shinagawa’s idea[3]. If not for this method, there could possibly be infinite values of \(\theta\) and \(\rho\) which would make the algorithm computationally expensive. In this case the \(W_0 = H_0 = D_{\text{max}}\) (as the Hough transform will be performed on the ring like window specified in Fig.4) which would make the discretization steps to be given by:

\[
d_\theta = \frac{3\pi}{4D_{\text{max}}}, \quad d_\rho = \frac{3}{4}
\]

2.3.2 Detecting Peaks

After performing the Hough transform, the next step would be to identify the peaks i.e., detecting the line segments within the search region. Let us assume, that \(C(\rho, \theta)\) represents the number of edge points satisfying the linear equation of Hough transform. The easiest way to calculate peaks would be set a threshold \((T_C)\), and extract all points that are above it. This can easily go wrong in the case of noisy images.

An alternate version to detect peaks is to identify and analyze butterfly patterns that arise in the vicinity of the peaks. In the paper, a simplified version of the butterfly evaluator suggested by Furukawa and Shinagawa was used to enhance the image. The enhanced image is given by:

\[
C_{\text{enh}}(\rho, \theta) = hw \frac{C(\rho, \theta)^2}{\int_{-h/2}^{h/2} \int_{-w/2}^{w/2} C(\rho + y, \theta + x) dx dy}, \quad (2)
\]
where $h$ and $w$ are the height and width of the rectangular region used for this enhancement, $C(\rho, \theta)$ represents the number of edge points satisfying the linear equation of Hough transform ($\rho = x \cos \theta + y \sin \theta$) and $C_{\text{enh}}$ is the enhanced hough image. Since $\rho$ and $\theta$ are quantized, the integral is computed through a convolution with a rectangular mask.

The local maxima of the enhanced image $C_{\text{enh}}(\rho, \theta)$ satisfying

$$C(\rho, \theta) \geq T_C$$

where $T_C$ is minimum number of pixels for the are stored as peaks.

An image showing the Hough transform of the test region with peaks is shown in Fig. 5. We can observe the 6 peaks, 4 of the rectangle bounded by the test region and 2 of the neighboring object peaks(edges).

![Fig. 5: Hough transform of the test region with detected peaks][2]

### 2.3.3 Detecting Rectangles

If $H_1 = (\rho_1, \theta_1), H_2 = (\rho_2, \theta_2), \ldots, H_m = (\rho_m, \theta_m)$ are the $m$ peaks that are detected in the specified region, we need to find the four peaks which satisfy the conditions specified in section 2.2.

Pairs of the peaks are scanned to satisfy the following conditions:

$$\Delta \theta = |\theta_i - \theta_j| < T_\theta,$$

$$\Delta \rho = |\rho_i + \rho_j| < T_\rho,$$

$$|C(\rho_i, \theta_j) - C(\rho_j, \theta_i)| < T_L \frac{C(\rho_i, \theta_j) + C(\rho_j, \theta_i)}{2}$$

$T_\theta$ corresponds to the angular threshold i.e., it determines if the peaks are parallel lines, which implies that $\theta_i \approx \theta_j$. $T_\rho$ is the distance threshold i.e., it determines if the lines are symmetric with respect to $\theta$ axis which implies that $\rho_i \approx -\rho_j$. $T_L$ determines the normalized threshold i.e., if the lines are approximately the same lengths ($C(\rho_i, \theta_j) \approx C(\rho_j, \theta_j)$).

To further extend the detection pattern, they specify that each pair of $H_i$ and $H_j$ which satisfy all the equations mentioned in this section will produce another extended peak

$$P_k = (\pm \xi_k, \alpha_k)$$

where

$$\alpha_k = \frac{1}{2}(\theta_i + \theta_j), \xi_k = \frac{1}{2}|\rho_i - \rho_j|$$

. The final step is to compare all pairs of extended peaks $P_k$ and $P_l$ and those that correspond to orthogonal pairs of parallel lines. These two extended peaks contain information about all the peaks which are potentially forming a rectangle. A rectangle is detected if:

$$\Delta \alpha = ||\alpha_k - \alpha_l| - 90^\circ| < T_\alpha$$

where $T_\alpha$ is an angular threshold that determines if the pairs of lines determined by $P_k$ and $P_l$ are orthogonal.

When a rectangle is detected, the intersection of the two pairs of parallel lines give you the vertices, $\alpha_k$ gives you the orientation, $2\xi_k$ and $2\xi_l$ give you the length of the sides.

### 2.3.4 Removing Duplicated Rectangles

Having multiple thresholds($T_\theta, T_\rho, T_\alpha$ and $T_L$) can lead to detection of duplicated rectangles for neighboring centers. When a sliding window is used to detect the centers of various objects in the image, there can be multiple centers which can be detected for the same object. Hough Transform will be performed applying the window and each of the centers, resulting in duplicating the rectangles i.e., the same rectangle with slightly different orientations is detected for each of the centers. This happens because of the small margin of error we leave when we compute the thresholds.

One way around this problem would be to set tighter(lower) thresholds. However, this may lead to wrongly identifying the actual rectangle.
A more efficient way is to calculate the error margin for each detected rectangle, and to choose the rectangle for which the error is the smallest. The paper suggests that there are five error measures related to each rectangle: Parallelism ($\Delta \theta_k, \Delta \theta_l$), Distance error ($\Delta \rho_k, \Delta \rho_l$) and Orthogonality error($\Delta \alpha$). The proposed error measure is given by:

$$E(P_k, P_l) = \sqrt{a(\Delta \theta_k^2 + \Delta \theta_l^2 + \Delta \alpha^2) + b(\Delta \rho_k^2 + \Delta \rho_l^2)}$$

where $a$ and $b$ are weights for angular and distance errors, respectively. However, measure of $\Delta \theta$ and $\Delta \rho$ are given in degrees and $\Delta \alpha$ in pixels. Visually, a difference of one pixel is more significant than a difference of one degree. To compensate the visual difference, the weight for distance error should be larger (typical values are $a=1, b=4$). The result can be seen in Fig.6.

Fig. 6: Results before and after error correction[2]

3 Results

3.1 Edge Detection

Several Edge detection techniques (Canny, Prewitt, Log, Sobel, Zerocross and Roberts) were tested and Canny and Prewitt were found to be the more efficient ones detecting edges more precisely. A low pass filters were used to smoothen the edges and the image was then dilated further and edges were detected for this particular image(Fig.8) using the ‘Canny’ edge detection method.

Fig. 7: Original Image

Fig. 8: Edges

3.2 Hough Transform implementation and rectangle detection

In this section, the Hough transform using the sliding window was applied and the various results are shown below in figures 9, 10 and 11.

As you can see the images below, there is lot or error. It initially detects the regions correctly. In fig.10 you can observe that a few rectangles are detected correctly on the left and right sides of the image, but there are a lot of squares detected because of the closeness of the edges and multiple peaks getting recognized in the process.

The results for the real life images are seen in figures 12,13 and 14. The noise in the image was initially reduced and the same algorithm was applied. Lot of other components were also detected along with the rectangles. There is still a lot of scope for improvement. Results for class 3 images are shown in 15,16,17 and are pretty much garbage.
Fig. 9: Potential regions in the image which can be rectangles overlapped with the original image (binary)

Fig. 10: All the rectangles detected without error correction

Fig. 11: Rectangles detected after error correction
Fig. 12: Potential regions in the image which can be rectangles overlapped with the original image (binary)

Fig. 13: All the rectangles detected without error correction

Fig. 14: Rectangles detected after error correction
Fig. 15: Potential regions in the image which can be rectangles overlapped with the original image (binary)

Fig. 16: All the rectangles detected without error correction

Fig. 17: Rectangles detected after error correction
4 ADDITIONAL WORK AND IDEAS

The proposed algorithm by Jung and Schramm is efficient only when there the rectangles in the images are sufficiently apart. This algorithm fails in the case of closely aligned images. An example for this failure is discussed in the original paper itself.

The ring window has several disadvantages. Observe Fig. 18. The Dmin and Dmax are the side of the smallest rectangle and diameter of the largest rectangle respectively.

Here, the ring window is shown as a the gray area. We observe how two other rectangles are also bound by the window in this case. The algorithm totally fails in such cases as it does not know what to do. The additional peaks detected will result in satisfying the equation more than once. This would result in wrongly detecting the rectangles which are about the same orientation and.

A solution to tackle this problem would be to use an elliptical window instead of a ring and also calculating the Dmin and Dmax of each object in the image individually. This might be computationally expensive but a more effective solution. A modified windowed transform is mentioned below.

4.1 Modified Windowed Transform algorithm

4.1.1 Edge detection

Instead of using Canny, Prewitt, Roberts, etc techniques to detect edges, we can simply dilate the image, adjust the intensity, remove noise, convert the image into binary and perform a few morphological operations to identify really good edges. All these edge detection result in losing a lot of information if the threshold is not decided properly. Results of my edge detection algorithm are shown in Figures 19, 20, 21.
4.1.2 Windowed Hough Transform

Instead of using the ring like Hough Transform where we manually input the Dmax and Dmin of the largest and the smallest possible rectangles in the image, we can identify all the objects in an image easily and calculate the longest and shortest side of the object individually on run time using Dijkstra’s algorithm. Also, to minimize the error further of the ring like window, we can chose an elliptical window and do the Hough transform of that.

This is not that tough to do, considering a circle is also a form of an ellipse. The equation of an ellipse is:

\[ \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \]  (3)

If a=b=r, then an ellipse becomes a circle with an equation \( x^2 + y^2 = r^2 \).

Let us assume that our rectangle sides are \( Rw \) and \( Rh \). We want an ellipse with the same proportions as rectangle. We need to keep the ratio of the ellipse \( \frac{a}{b} \) same as the ratio of the rectangle sides \( \frac{Rw}{Rh} \). This gives us another equation:

\[ \frac{a}{b} = \frac{Rw}{Rh} \]  (4)

Solving 3 and 4, we have

\[ a = b \left( \frac{Rw}{Rh} \right) \]
\[ \frac{Rh}{2b} + \frac{Rh}{2b} = 1 \]
\[ Rh = \sqrt{2b} \]

Solving for both \( a \) and \( b \) we have

\[ a = \frac{Rw}{\sqrt{2}}, b = \frac{Rh}{\sqrt{2}} \]

Here is an example solution shown in Fig 22.

This window eliminates detection of many undesired and additional edges making the search area smaller. Also, calculation of \( Rw \) and \( Rh \) real time rather than manually giving a Dmin and Dmax value improves the efficiency of the algorithm to a great level.

The modified algorithm was implemented to a certain extent and hasn’t been completed due to time constraints. Overall, with a little modification to Jung and Schramm’s algorithm we can produce a very efficient algorithm for edge detection which might be expensive computationally but very efficient.

4.2 Drawbacks

A major drawback is that, these algorithms do not account for overlapped rectangles. These can only detect individual rectangles that are spaced apart. Future work would be to implement such an algorithm which could detect rectangle in the case of an overlapped image as well.

REFERENCES