Accurate Line Detection by Adjusting Hough Transform Threshold Adaptively

Yan Ma  Junning Wang  Di He
Department of Communication Engineering
Xidian University
Xi’an, China
my1219_love@126.com

Jinbo Pang
Institute of Photoelectronics
Nankai University
TianJin, China

Abstract — A method to adjust the Hough transform threshold automatically is proposed in this paper. Straight lines are divided into two categories, parallel lines and intersection lines. Firstly we adjust the threshold roughly depending on the number of the needed lines. Then, the threshold are transformed precisely according to different characteristics of the two kinds of straight lines. Typically, this algorithm is applied to discern the runway in the aerial images of airports and recognize the bridge. Simulation results show that the algorithm requires short time and has a good anti-noise performance. Furthermore, it can solve the problems of “false peak” and “false negative”.

Keywords — Hough transform; Line detection; Detection threshold; Adaptive adjustment of the threshold

I. INTRODUCTION

A number of man-made targets can be described as straight lines, such as buildings, airport, major roads and so on. Thus, the extraction of straight lines is an important component for the target recognition and scene analysis systems[1]. And the Hough transform is one of the most commonly used methods to detect straight lines.

When solving the detection problem in image space, if we transform it into the parameter space, we shall recognize the line easily by detecting the peak after counting and adding up the parameters. This process is the Hough transform[2]. The Hough transform has good anti-noise performance and it makes short lines in the same straight line connect to one line. The disadvantage is that the parameters are difficult to choose and complex to compute[3].

Xu[4] proposed a new algorithm the randomized Hough transform (RHT). The system meets basic requirements of efficient and reliable operation with the reasonable algorithm to reduce the amount of computation and improve the computational speed. But the sample space of RHT is all the edge points, so it is only suitable for images with less edge pixels. For some invalid samples and accumulation problem of RHT, Zhang[5] proposed a detection method by using the edge gradient direction rather than the edge points as the sample space. The random sample space becomes smaller, thus the speed is improved. In this approach, the author does not take into account the nature of straight lines of the edge image. This may makes points with the same straight line be assigned to different groups, but points with different straight lines be assigned to the same group. Chen[6] took into account this defect. In order to get effective grouping of edge points, he found a matrix and did convolution using this matrix and the edge image. Then each image edge point is marked with a pseudo-direction. Under the guidance of pseudo-directions, only the edge points in the same direction are chosen as the sample space. The sample space is small and the grouping is accurate. This is the algorithm of straight line detection based on pseudo-direction. However, to use this algorithm we need to estimate the approximate direction of the straight lines in the image. If the estimated direction is incorrect, there will be confusion straight lines shown in the result images.

These methods mentioned above use a given threshold to determine the estimated parameters. However, the characteristics of Hough transform make the estimated parameters be much more than the real parameters and many estimated parameters come from the same straight lines[7]. So it does not conform to the reality to select straight lines detected by the estimated parameters directly as the output[7]. Take into account the defects of straight-line detection using given threshold and inspired by the concept of pseudo-orientation, we propose a method in this paper, which is used to detect straight lines by adjusting the Hough transform threshold automatically. Straight lines are divided into two categories, parallel lines and intersection lines. On this basis, effective straight lines are detected, combined with features of required lines and image. This algorithm is applied to detect more than one straight lines, but only need to do once Hough transform. Thereby improving the speed. This algorithm can be used to discern the runway in the aerial images of airports and recognize the bridge. In the experiment, the algorithm proposed in this paper is compared with the algorithm of straight line detection based on pseudo-orientation and the algorithm of randomized Hough transform. The results show that the algorithm proposed in this paper takes less time to detect straight lines accurately. Also it is in a good anti-noise performance.

supported by the 111 Project(B08038)
II. ANALYSIS OF THE IMPORTANCE OF ADJUSTMENT FOR HOUGH TRANSFORM THRESHOLD

The mapping form of a straight line in the cartesian coordinates system is \((ρ, θ)\) in the parameter space. Rho is the distance from the origin to the straight line. Theta is the vector angle of perpendicular from the origin to the straight line. \(ρ\) and \(θ\) are illustrated in Fig.1.

![Hough transform schematic](image)

Figure 1. Hough transform schematic

Hence, the corresponding relationship can be expressed as this formula:

\[
ρ = x \cdot \cos θ + y \cdot \sin θ, \quad θ ∈ [0, π]
\]

The straight line is changed into a sine curve in the polar coordinates, via the Hough transform. It is shown that points in the same straight line mapped to different sine curves in polar coordinates and these sine curves share a public point\(^8\). And this is a powerful proof that one point in the polar coordinates corresponds to one straight line in the cartesian coordinates. And they are one to one mapping.

In order to detect straight lines in a picture, polar coordinates \((ρ, θ)\) can be quantified into many small units. According to coordinates \((x,y)\) of each edge point, the corresponding value of \(ρ\) is computed by the step-size of a small grid within the range \(θ ∈ [0, π]\). If the value is restrained within a small grid, accumulation counter for this small lattice increases by one\(^8\). After the edge points are all processed in this method, we obtained accumulation matrix \(acc\). The accumulation value for each point \((ρ, θ)\) is expressed as \(acc(ρ, θ)\). The small units are tested. If the accumulation values of several points \((ρ, θ)\) agree with this formula \(acc(ρ, θ) > T\) we consider the mapping lines of \((ρ, θ)\) as the required lines.

\(T\) is short for the threshold. If the accumulation values of \((ρ, θ)\) agree with the condition that \(acc(ρ, θ) > T\), we think the mapping lines of the points exist in the picture. If \(T\) is smaller, more points match the condition and thus more lines are marked, with confusion lines shown in the results. In contrast, if \(T\) is larger, there will be less points meet the condition, which results in the loss of several lines. It is paramount to choose the compromising value for \(T\), which can both improve transform efficiency and achieve higher accuracy.

III. ADAPTIVE ADJUSTMENT OF THE HOUGH TRANSFORM THRESHOLD

In practice, if the straight lines are not parallel they must be intersecting.

As is known, the same number of straight lines in Cartesian coordinates system corresponds to the same number of points in the polar coordinates with one-to-one mapping relationship. When all the theta are equal, while rho are different, the mapping straight lines of these points are parallel with each other. Similarly, if more than two different rho are among \(n\) points in the polar coordinates, the mapping straight lines of these points are intersection lines. The maximum of accumulation matrix \(acc\) is expressed as \(acc_{max}\). \(numel(ρ, θ)\) is the accumulation number of \((ρ, θ)\). When \(numel(ρ, θ)\) equals \(n/2\), points are written as \((ρ_m, θ_m)\) and the maximum of \(acc(ρ_m, θ_m)\) denoted by \(acc_m\).

\(T_0\) is the initial value for \(T\). In this paper, \(T_0\) is computed by the following equation.

\[
T_0 = \frac{acc_{max} + acc_m}{2}
\]

Up to now, \(T_0\) is a middle value. \(T_0\) is initially evaluated for the middle value, thereby decreasing the adjustment range compared with using the initial value of minimum or maximum.

A. Required lines are parallel straight lines

1) Adjust \(T_0\) roughly depending on the number of required lines

   (1) If \(T_0\) makes \(numel(ρ, θ)\) less than \(n\), it shows that \(T_0\) is larger. \(T_0\) is reduced by \(ΔT\) step by step. Since the smallest unit of a digital image is 1 pixel, we choose 1 as the value of \(ΔT\) for higher accuracy. \(T_0\) decreases, until \((ρ, θ)\) agrees with the condition \(acc(ρ, θ) > T\), content the formula \(numel(ρ, θ) ≥ n\).

   (2) If \(T_0\) makes \(numel(ρ, θ)\) more than \(n\), it shows that non-target lines appear.

   Definition 1 Since there are several value for theta within the saved \((ρ, θ)\), we define the theta that appears most frequently as mainstream \(θ_s\), denoted by \(θ_s\). Other thetas are defined as non-mainstream \(θ_s\), expressed as \(θ_0\).

   All \(θ\) are evaluated, to save \(θ_0\) and note down \(θ_s\). Then \((ρ, θ_s)\) is removed. For \(θ_s\) is written down, so \((ρ, θ_s)\) can be removed firstly in the next adjustment process.

   When all theta in the accumulation matrix are \(θ_0\), there may be two cases. One case is that \(numel(ρ, θ)\) is not less than \(n\), then increase \(T_0\) by \(ΔT\) step by step and \(ΔT\) equals 1. Until \((ρ, θ)\) that meent the condition \(acc(ρ, θ) > T\), make
numel(ρ, θ) not more than n. The other case is that numel(ρ, θ) is less than n, do nothing.

2) Adjust $T_0$ precisely according to the case parallel straight lines have the same $\theta$

Select $\theta$ according to $\theta_0$ and remove $(\rho, \theta_0)$ within the saved $(\rho, \theta)$. In order to remove $(\rho, \theta_0)$ firstly in the next adjustment process, note new $\theta_0$ down before. If numel$(\rho, \theta)$ less than n, reduce $T_0$ by $\Delta T$ step by step, until the points that agree with the condition $acc(\rho, \theta) > \Gamma$, make numel$(\rho, \theta)$ not less than n.

3) Adjust $T_0$ precisely according to the condition parallel straight lines have different $\rho$

Definition 2 Consider $\rho_0$ in one point $(\rho_0, \theta_0)$ as the center of a little neighborhood, we define other $\rho$s which are in the neighborhood of $\rho_0$ as interference values of $\rho_0$. The neighborhood with radius $r$ and its area is very small. We define the center $\rho_0$ as the mainstream $\rho$, denoted by $\rho_1$. Interference values are defined as non-mainstream $\rho$, expressed as $\rho_i$.

In this paper, $r$ is equal to 3. If two points with the same theta and the difference between their rho is within 3, the mapping straight lines of them are considered as one straight line. Two points (226,15) and (223,15), their mapping straight lines are shown on Fig.2. They look like one line approximately. So making $r$ equal 3 is reasonable. Of course, we can choose the value of $r$ according to the actual situation.

![Figure 2](image)

Figure 2. two mapping lines of two points

$(\rho_i, \theta)$ will be removed. after we save $\rho_1$ and note down $\rho_i$. Since new $\rho_i$ is written down, we can remove $(\rho_i, \theta)$ firstly in the next adjustment process.

If there is $\rho_i$, after $(\rho_i, \theta)$ are removed, the number of $(\rho_i, \theta)$ will be smaller than n. Reduce $T$ by $\Delta T$ step by step, until all $\rho$s in the cumulation matrix are $\rho_i$. Up to now, all the $(\rho, \theta)$ are marked as $(\rho_i, \theta)$ and the number of $(\rho_i, \theta)$ is equal to n.

Now if the number of $\theta_0$ is less than n, we consider $\theta$ near $\theta_0$, as $\theta$ and the difference between them is smaller than 2. Because the image pixels are so small that the distance is reasonable. Reduce $T$ by $\Delta T$ step by step, until $(\rho, \theta)$ agree with that $acc(\rho, \theta)$ is more than $\Gamma$ and the number of $(\rho, \theta)$ equals n.

B. Required lines are intersection straight lines

1) Adjust $T_0$ roughly depending on the number of required lines

(1) If $T_0$ makes numel$(\rho, \theta)$ less than n, it shows that $T_0$ is larger. Reduce $T_0$ by $\Delta T$ step by step, until $(\rho, \theta)$ that agree with the condition $acc(\rho, \theta) > \Gamma$, make the equation numel$(\rho, \theta) \geq n$. $\Delta T$ is equal to 1.

(2) If $T_0$ makes numel$(\rho, \theta)$ more than n, it shows that $T_0$ is smaller. $T_0$ increase $\Delta T$, until $(\rho, \theta)$ that agree with the condition $acc(\rho, \theta) > \Gamma$, make the formula numel$(\rho, \theta) \leq n$ hold.

2) Adjust $T_0$ precisely according to the condition intersection straight lines have different $\theta$

Definition 3 Consider $\theta_0$ in one point$(\rho, \theta_0)$ as the center of a little neighborhood, we define other $\theta$s which are in the neighborhood of $\theta_0$ as interference values of $\theta_0$. The neighborhood with radius $r$ and its area is very small. We define the center $\theta_0$ as mainstream $\theta$, denoted by $\theta_0$. The interference values are defined as non-mainstream $\theta$, expressed as $\theta_i$. Up to now, $(\rho, \theta)$ is expressed as $(\rho, \theta_i)$.

In this paper, $r$ is equal to 2. If the difference between two thetas is within 2, they are considered as the same theta. If their rhos are approximately equal, we think their mapping straight lines are overlap lines. Two points (3,56) and (4,58), their mapping straight lines shown in Fig.3. They look like one same line approximately. Hence, it is reasonable to make $r$ equal to 2. Of course, we can choose the value of $r$ according to the actual situation.

![Figure 3](image)

Figure 3. two mapping lines of two points

If several rhos in $(\rho, \theta_i)$ are not $\rho_0$, it means that there are numbers of parallel lines in these intersecting lines. Detect these parallel straight lines, in accordance with the method described earlier. Then detect intersecting lines altogether.

If there are $\theta_i$, the numel$(\rho, \theta_i)$ will be less than n. $T$ is decreased, until all the marked theta are $\theta_i$. Now all points are expressed as $(\rho, \theta_i)$, and the number of $(\rho, \theta_i)$ is equal to n.

C. Analysis of the time complexity

$N$ is the number of required lines. Via edge detection, we obtain $M$ edge points in the image. $T_{rand}$ represents the threshold of the algorithm of straight line detection based on pseudo-direction and RHT, $M_1$ is the number of the edge points with a pseudo-direction. The time complexity of RHT$^{[5]}$ is $O(M \times T_{rand} \times N)$. The time complexity of the algorithm proposed by chen$^{[6]}$ is $O(M \times T_{rand} \times N)$. The time complexity of the algorithm proposed in this paper is expressed as $O(M)$. From the theoretical analysis, we can see that the algorithm proposed in this paper will take less time.
IV. EXPERIMENT AND ANALYSIS

A. Experiments of detection for parallel lines

We apply the algorithm proposed in this paper to the experiments of a large number of aerial images and analyze the performance of the algorithm. The computation is carried out in the software of Matlab on a computer platform is Pentium (R) IV 2.93G (memory, 512M). The results are ideal. Fig.4A is an aerial image of an airport runway, with the size of 512 × 443 pixels. In order to analyze the anti-noise performance of this algorithm, we add noise to blur the image in the experiments. Fig.4F is an image blurred with Gaussian noise. The Gaussian noise with mean is 0, and variance is 0.1. Fig.4K is an image with salt and pepper noise whose density is 30%. This paper introduces the comparison analysis of the algorithm of straight line detection based on pseudo-orientation, the algorithm of random Hough transform and the algorithm proposed in this paper.

The algorithm proposed in this paper is called as Algorithm 1. The algorithm of straight line detection based on pseudo-orientation is named Algorithm 2. The algorithm of random Hough transform is Algorithm 3.

![Figure 4. experiment images of airport runway detection](image)

<table>
<thead>
<tr>
<th>Table I.</th>
<th>Table Column Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Head</td>
<td>Table Column Head</td>
</tr>
<tr>
<td>algorithm</td>
<td>Maximum sample space</td>
</tr>
<tr>
<td>Comparison of three algorithms</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td></td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>Algorithm 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II.</th>
<th>Table Column Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Head</td>
<td>Table Column Head</td>
</tr>
<tr>
<td>algorithm</td>
<td>Maximum sample space</td>
</tr>
<tr>
<td>Comparison of three algorithms</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td></td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>Algorithm 3</td>
</tr>
</tbody>
</table>

Fig.4B, Fig.4G and Fig.4L are the edge detection images of Fig.4A, Fig.4F and Fig.4K respectively. We use Canny method to detect the image edge in this paper. It is easy to see that interference edge is more in Fig.4L than that in Fig.4B and Fig.4G. Straight lines from Fig.4B, Fig.4G and Fig.4L are detected by Algorithm1, and the result images are Fig.4C, Fig.4H and Fig.4M respectively. The result images show that using Algorithm1 to detect straight lines, the detection lines are not affected by noise. Fig.4D, Fig.4I and Fig.4N are results to recognize straight lines by Algorithm2 and the results are good. We apply Algorithm3 to detect straight lines in the three pictures, the results are shown in Fig.4E, Fig.4J and Fig.4O. It can be seen that in Fig.4O there are confusion straight lines. In the algorithm proposed in this paper, we pay attention to the situation and make sure that there will be no confusion lines.

Next, we analyze the speed and anti-noise performance of Algorithm1 by the specific experimental data. These data are from the experiments to recognize parallel lines by Algorithm1, Algorithm2 and Algorithm3.

![A  original image](image) B  edge of original image C  lines' detection by Algorithm 1 D  lines' detection by Algorithm 2 E  lines' detection by Algorithm 3

![F  gaussian noise image](image) G  edge of Fig.E H  lines' detection by Algorithm 1 I  lines' detection by Algorithm 2 J  lines' detection by Algorithm 3

![K  salt and pepper noise image](image) L  edge of Fig.K M  lines' detection by Algorithm 1 N  lines' detection by Algorithm 2 O  lines' detection by Algorithm 3


Table III.

<table>
<thead>
<tr>
<th>Table Head</th>
<th>Table Column Head</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>algorithm</td>
</tr>
<tr>
<td>Comparison of three algorithms</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td></td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>Algorithm 3</td>
</tr>
</tbody>
</table>

Table I, Table II and Table III show the time-consuming situations of the detection process of Fig.4A, Fig.4F and Fig.4K respectively, by Algorithm1, Algorithm2 and Algorithm3. The data is the average data from 10 tests. From Table I, we can see that compared with Algorithm2 the rate of Algorithm1 increases 5 ~ 6 times in no-noise cases and compared with Algorithm3 the rate of Algorithm1 increased by about 30 times. The performance of high efficiency and good anti-noise of this Algorithm1 is shown better in Table II and Table III. Since there are more edge points in Fig.4G and Fig.4L, it takes more time. From the tables, it can be seen that it needs only about 1 second more to detect lines in Fig.4G and Fig.4L by Algorithm1. The impact of noise on the time-consuming of Algorithm1 is very small. But there is greater impact to the time-consuming by using other two algorithms.

B. Experiments of detection for intersection lines

Fig.5A is one bridge with the size of 509 × 385 pixels. Fig.5F is a gaussian noise blurred image. The gaussian noise with mean 0, and variance equal 0.1. Fig.5K is an image with salt and pepper noise whose density is 30%.

The algorithm proposed in this paper is called as Algorithm1. The algorithm of straight line detection based on pseudo-orientation is named Algorithm2. The algorithm of random Hough transform is Algorithm3.

![Figure 5](image_url)

Fig.5B, Fig.5G and Fig.5L are the edge detection images of Fig.5A, Fig.5F and Fig.5K respectively. The result images Fig.5C, Fig.5H and Fig.5M show that the detection for intersection lines by Algorithm1 is as good as the detection results of parallel lines. Fig.5D, Fig.5I and Fig.5N are the detection results by Algorithm2. Shown on Fig.5I are two lines and they look like one same line. Another required line is missed. The same problem is shown on Fig.5E, Fig.5J and Fig.5O, the results by Algorithm3. We take note of it, the interference factors are excluded in Algorithm1. So there will not be overlap lines, nor missing lines.

Next, we analyze the speed and anti-noise performance of Algorithm1 by the specific experimental data. These data are the average data that from 10 experiments to detect intersection lines by using Algorithm1, Algorithm2 and Algorithm3.

Table IV.

<table>
<thead>
<tr>
<th>Table Head</th>
<th>Table Column Head</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>algorithm</td>
</tr>
<tr>
<td>Comparison of three algorithms</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td></td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>Algorithm 3</td>
</tr>
</tbody>
</table>

Table V.

<table>
<thead>
<tr>
<th>Table Head</th>
<th>Table Column Head</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>algorithm</td>
</tr>
<tr>
<td>Comparison of three algorithms</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td></td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>Algorithm 3</td>
</tr>
</tbody>
</table>
Table VI.

<table>
<thead>
<tr>
<th>Table Head</th>
<th>Table Column Head</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum sample space</td>
</tr>
<tr>
<td>Comparison of three algorithms</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td></td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>Algorithm 3</td>
</tr>
</tbody>
</table>

Table4, Table5 and Table6 show the time-consuming situations of the detection process of Fig.5A, Fig.5F and Fig.5K respectively, by Algorithm1, Algorithm2 and Algorithm3. These data show that compared with Algorithm2 and Algorithm3, Algorithm1 has higher speed and better anti-noise performance.

V. CONCLUSION

A method suitable to detect numbers of straight line is presented in this paper. Firstly, we adjust the detection threshold depending on the features of required straight lines. Then we choose the ideal target point, so there will be neither overlap lines, nor missing lines. Also, this mind is also suitable for other methods of line detection. Compared with the algorithm of straight line detection based on pseudo-orientation and the algorithm of random Hough transform, the algorithm proposed in this paper needs less amount of computation and improves the speed. Finally, the results show that the algorithm has a good ability to distinguish lines in the noise blurred images.

ACKNOWLEDGEMENT

This work has been supported by the 111 Project(B08038). We really appreciate the kindly discussion with teachers in our lab. And the help from the sponsor is greatly acknowledged.

REFERENCES