Randomized Generalized Hough Transform for 2-D Grayscale Object Detection *

Ping-Fu Fung, Wing-Sze Lee and Irwin King †

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
E-mail: {pffung, wslee, king}@cs.cuhk.edu.hk

Abstract

This paper proposes a new algorithm for 2-D object detection called Randomized Generalized Hough Transform (RGHT). It combines the Generalized Hough Transform (GHT) with the Randomized Hough Transform (RHT). Our algorithm can detect arbitrary objects of various scales and orientations in graylevel images. We also demonstrate RGHT's advantage of high speed, low storage requirement, high accuracy and arbitrary resolution through comparison with other related algorithms.

1. Introduction

One of the key issues in image processing is to extract interested objects from an image. Hough Transform (HT) [2] is a classical algorithm for extracting lines from a binary image. Another improved algorithm is the Generalized Hough Transform (GHT) [1] which can detect arbitrary object in a grayscale image. With little modifications, the algorithm can detect objects of various scales and orientations at the expense of more computation time. A faster algorithm called Randomized Hough Transform (RHT) [4] detects various analytical geometric shapes in binary images by using probabilistic method. Since random sampling is used, the algorithm works quite fast with low storage requirement.

Here, we present a new algorithm for object detection. It combines the advantages of both GHT and RHT; hence it is named Randomized Generalized Hough Transform (RGHT). It works on grayscale images and detects arbitrary grayscale object with various scales and orientations, like GHT. Moreover, it is based on probabilistic method as in RHT, so it has reduced time and space complexities, high accuracy and arbitrary resolution.

*This work is supported in part by RGC Earmarked Grant #221500620 and Industry Grant #2427 01300 of Hong Kong.

† The contacting author.

2. Background

2.1. The Hough Transform

The HT is based on transformation of image points' coordinates into a 2-D line parameter space. HT is slow due to a large number of lines need to be plotted in the parameter space. Moreover, it can only work on binary images. Nevertheless, a slight modification can be made for detecting analytical shapes other than line, for example, circle, ellipse, etc. at the expense of greater storage complexity (due to larger parameter space) and heavier computation (more complex equations are involved).

2.2. The Generalized Hough Transform

The GHT detects arbitrary object, not limited to analytical geometric ones, in a grayscale image. It uses the gradient information to generate a lookup table, called R-table for shape detection. Original GHT is only sensitive to a specific size and orientation of an object. Although performing GHT iteratively with augmented R-table is a remedy, the resulting computation effort increases drastically with bad accuracy and resolution.[1] We analyse time and space complexities of GHT in a later section.

2.3. The Randomized Hough Transform

The RHT is based on probabilistic method. Since sampling is used, a small subset of points is examined thus the computation time and storage are reduced. It guarantees infinite and continuous parameter space. However, current versions of RHT only work on binary images so that a preparation step of thresholding or filtering has to be applied to grayscale images beforehand. Also, it only detects analytical geometric shapes rather than arbitrary ones.

[1] Scaling the offsets in the R-table for detecting scaled objects, shifting table entries for detecting rotated objects

1015-4651/96 $5.00 © 1996 IEEE
Proceedings of ICPR '96

3. The RGHT

3.1. Basic idea

The RGHT combines both GHT and RHT, thus it shares the advantages of the two algorithms. Before showing how the algorithm works, the notations employed in the following context are listed:

Px_i, Py_i	x, y-coordinates of pt. $P_i(x_i, y_i)$
$O(x,y), \widehat{O}(x,y)$	actual and est. anchor points
$S(x,y), \widehat{S}(x,y)$	actual and est. scaling factors
$\Delta_i(x_i, y_i)$	offset of $P_i(x_i, y_i)$ from $O(x,y)$
$\Delta h(x,y), \Delta v(x,y)$	hori. and vert. local gradient
θ_i	gradient angle at $P_i(x_i, y_i)$

Given a template image, in which the interested grayscale object resides, we apply a gradient operator to the image to produce a R-table, just as in GHT. The gradient operator makes use of two filtering masks: the horizontal mask and the vertical mask, as shown in **Fig. 1**. By performing convolution on the template image with these two masks independently, we obtain two gradient difference maps. As illustrated in **Fig. 2**, local gradient at each point in the template image is given by:

$$\theta = \arctan\left(\frac{\Delta v(x,y)}{\Delta h(x,y)}\right) \tag{1}$$

In the template image, we choose an anchor point $O(x,y)$ as

$$\begin{bmatrix} -1 & -2 & -3 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -3 & 0 & 3 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figure 1. Horizontal and Vertical masks

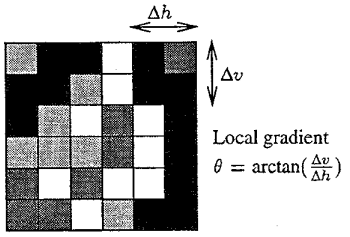


Figure 2. Applying gradient operator

a reference origin point. In order to reduce calculation error, it had better to pick the center of the minimum bounding box of the template object as anchor. Such choice reduces the range of the offset values in the R-table, thus the relative

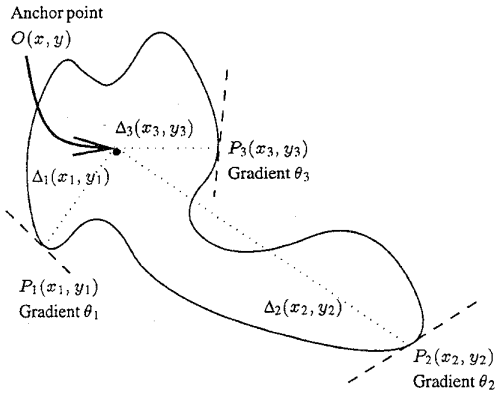


Figure 3. Building R-table

error is minimized. From any point $P_i(x_i, y_i)$ in the image, there is an offset to the anchor point:

$$\Delta_i(x_i, y_i) = O(x,y) - P_i(x_i, y_i)$$

Combining the local gradient and offset, we form the R-table. The R-table indicates the offset to the anchor point from other image points with various local gradients. **Fig. 3** illustrates how the table is created and **Table 1** shows the structure of the R-table. Note that the R-table is actually a one-to-many mapping and empty entries are allowed.

Gradient in Degree	List of Offset Pairs
0	$\Delta_1(x_1, y_1), \Delta_2(x_2, y_2)$
1	$\Delta_3(x_3, y_3)$
2	Nil
⋮	⋮
358	$\Delta_m(x_m, y_m), \ldots, \Delta_n(x_n, y_n)$
359	Nil

Table 1. An example of the R-table

After generating the R-table, we proceed to the object detection process.[2] Given a target image, our task is to locate the interested object and tell the scale of the detected object. In other words, we are trying to find out $(\widehat{O}x, \widehat{O}y, \widehat{S}x, \widehat{S}y)$ in our 4-D parameter space $x_{pos} \times y_{pos} \times x_{scale} \times y_{scale}$. Given any point $P_i(x_i, y_i)$ in the target image, we calculate the local gradient θ_i by **Eq. 1**. Using θ_i to index into the R-table gives us the offset $\Delta_i(x_i, y_i)$. Then the following

[2] Prepare and save the R-table for each template image once is enough, e.g. one R-table for circle (ellipse included due to stretched scaling), one for pentagon, etc.

relation can be set up:

$$(\widehat{O}x, \widehat{O}y, \widehat{S}x, \widehat{S}y) = (Px_i, Py_i, 0, 0) + \widehat{S}x \cdot (\Delta x_i, 0, 1, 0) + \widehat{S}y \cdot (0, \Delta y_i, 0, 1)$$

The above linear combination relates $P_i(x_i, y_i)$, $\Delta_i(x_i, y_i)$, $\widehat{O}(x, y)$ and $\widehat{S}(x, y)$ in our 4-D parameter space. For computational purpose, we rewrite it in vector notation:

$$(\widehat{O}x, \widehat{O}y) = (Px_i, Py_i) + (\widehat{S}x, \widehat{S}y) \cdot (\Delta x_i, \Delta y_i)$$

In the target image, pick any two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, as shown in **Fig. 4**, we have the following equations:

$$(\widehat{O}x, \widehat{O}y) = (Px_1, Py_1) + (\widehat{S}x, \widehat{S}y) \cdot (\Delta x_1, \Delta y_1) \quad (2)$$

$$(\widehat{O}x, \widehat{O}y) = (Px_2, Py_2) + (\widehat{S}x, \widehat{S}y) \cdot (\Delta x_2, \Delta y_2) \quad (3)$$

Equating **Eq. 2** and **Eq. 3**, we get

$$\widehat{O}x = Px_1 + \widehat{S}x \cdot \Delta x_1 = Px_2 + \widehat{S}x \cdot \Delta x_2 \quad (4)$$

$$\widehat{O}y = Py_1 + \widehat{S}y \cdot \Delta y_1 = Py_2 + \widehat{S}y \cdot \Delta y_2 \quad (5)$$

Rearranging terms and obtain:

$$\widehat{S}x = \frac{Px_2 - Px_1}{\Delta x_1 - \Delta x_2} \quad (6)$$

$$\widehat{S}y = \frac{Py_2 - Py_1}{\Delta y_1 - \Delta y_2} \quad (7)$$

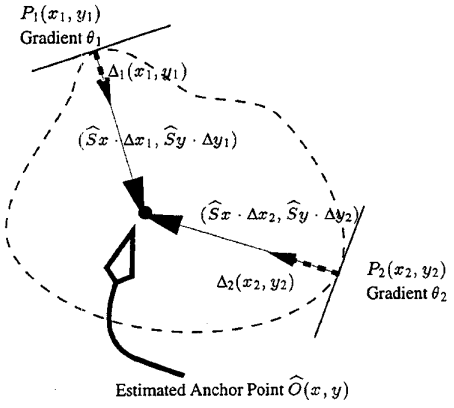


Figure 4. Estimating $O(x, y)$ and $S(x, y)$

After evaluating **Eq. 6** and **Eq. 7**, we find $\widehat{S}(x, y)$. Then, substitute the result into **Eq. 4** and **Eq. 5**, we get $\widehat{O}(x, y)$. Finally, we obtain the estimated anchor point $\widehat{O}(x, y)$ and the estimated scale $\widehat{S}(x, y)$ of the template object in the image.

By picking randomly two sample points from the target image, we obtain a point $(\widehat{O}x, \widehat{O}y, \widehat{S}x, \widehat{S}y)$ in the parameter space. After a number of iterations and accumulate the result, peaks in the accumulation indicate the location $O(x, y)$ and scale $S(x, y)$ of the template object in the target image. Undergoing a peak seeking process, we can find out the parameters of the detected objects.

To detect rotated objects, we need to modify the R-table and then re-apply the algorithm on the target image. The modification is simple: shift the entries in the R-table. In this way, we detect rotated objects at the expense of spending more computation time. Iterating for different angles, template object with different orientations can be detected.

3.2. Procedure and analysis

Algorithm 1 Generate-R-Table(in: Template; out: R-Table)

▷ *Template image dependent, target image independent*
1 Horizontal-gradient ← Convolve(Template, Horizontal-mask)
2 Vertical-gradient ← Convolve(Template, Vertical-mask)
3 Anchor ← Center of Min-bounding-box(Template)
4 **for** $(x, y) \in$ Template
5 $\theta \leftarrow \arctan(\frac{\text{Horizontal-gradient}_{(x,y)}}{\text{Vertical-gradient}_{(x,y)}})$
▷ *R-table is an array of linked-list*
6 List-insert(R-Table[θ], Anchor $- (x, y)$)
7 **end**

Algorithm 2 RGHT(in: Image, R-Table, Iteration, Resolution, Threshold; out: Result-set)

▷ *Template image independent, target image dependent*
1 Horizontal-gradient ← Convolve(Image, Horizontal-mask)
2 Vertical-gradient ← Convolve(Image, Vertical-mask)
3 Solution-set ← ϕ ▷ *The accumulation set*
4 **for** i ← 1 to Iteration
5 Generate $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ randomly
6 Calculate θ_1 and θ_2 of P_1 and P_2 ▷ *By* **Eq. 1**
7 Lookup $\Delta_1(x_1, y_1)$ and $\Delta_2(x_2, y_2)$ from R-Table by θ_1, θ_2
8 Calculate $\widehat{O}(x, y)$ and $\widehat{S}(x, y)$ ▷ *By* **Eq. 4** *to* **Eq. 7**
9 **if** $\exists$ Sol $\in$ Solution-set
10 s.t. $|(\widehat{O}x, \widehat{O}y, \widehat{S}x, \widehat{S}y) - \text{Sol}| \leq$ Resolution **then**
▷ *Check if there is any resemble solution in accumulation set*
11 Merge $(\widehat{O}x, \widehat{O}y, \widehat{S}x, \widehat{S}y)$ with Sol
▷ *Take average and increase hit rate of Sol*
else
12 Solution-set ← Solution-set $\cup$ $(\widehat{O}x, \widehat{O}y, \widehat{S}x, \widehat{S}y)$
▷ *Insert the new solution into the accumulation set*
13 **end**
14 Result-set ← ϕ ▷ *The result set*
15 **for** Sol $\in$ Solution-set
16 **if** hit rate of Sol $\geq$ Threshold **then**
▷ *Extract result from accumulation set*
17 Result-set ← Result-set $\cup$ Sol
18 **end**

In the two algorithms listed above, there are three important parameters can be tuned:

Iteration The number of iterations (i.e. the sampling size) controls the number of sample pairs taken. If it is set too

large, more computation effort is required. However, result is inaccurate if it is not large enough because out-lier will contribute relatively higher proportion. Generally, it depends on the size of the image: number of iterations $\propto x_{size} \times y_{size}$. Its value may be dynamically determined. Whenever a solution with relatively high hit rate is encountered, the process can be stopped. [3] gives more information on stopping criteria of RHT and the analysis is applicable to all sorts of RHT based algorithms.

Resolution It determines when two solutions are considered as close. If two solutions are resemble ($|S_1 - S_2| \leq$ Resolution), they are merged to form a single solution. Small resolution results in no merging at all and a large number of trivial solutions will be reported. Large values cause many solutions to be merged together; thus, they will get high hit rate with relatively lower resolution.

Threshold It controls the peak seeking process. Low threshold will cause a lot of trivial solutions to be found while peaks may be missed with high threshold value. We may define the threshold in terms of the maximum hit rate, for example, $\frac{1}{4}$ of the value.

In order to make the algorithm work more efficiently, we suggest some special data structures. The R-table may be stored as an array of linked-list. For the 4-D parameter space, (Ox, Oy, Sx, Sy), the storage required is in the order of $O(x_{size} \cdot y_{size} \cdot \#x_{scale} \cdot \#y_{scale})$. However, due to application of random sampling, the result is sparse enough that hashing technique helps to reduce the storage. To facilitate high speed searching of resemble solutions and merging them, the 4-D parameter space is reduced into a 4-D hash table. Each bucket of the hash table is implemented by a list. This data structure is very space saving and efficient for storing the result without loss of accuracy or resolution.

3.3. Characteristics and advantages

Works on Grayscale Images Without thresholding or filtering, RGHT works on the original image with greater details.

Detects Arbitrary Object RGHT detects even free hand drawn shapes and gray filled objects.

Finds Scaled/Stretched Instances in a Single Run RGHT does not need to do iteration but rather detects all scaled instances at one time.

High Speed by Random Sampling By random sampling, RGHT does not need to scan the whole image.

High Accuracy by Equation Solving Object location and scale are detected by equation solving and parameters are not limited to integral discrete values.

Arbitrary Resolution The solution merging process allows adjustment of the resolution of object detection.

3.4. Comparison between RGHT and other Hough Transforms

Algorithm	HT	GHT	RHT	RGHT
Image type				
Binary	√	√	√	√
Grayscale	×	√	×	√
Target				
Line	√	×	√	×
Circle	√	√	√	√
Parametric	√(slow)	√	√(slow)	√
Arbitrary	×	√	×	√
Speed	slow	slow	fast	fast
Storage	high	high	low	low
Accuracy	medium	medium	high	high
Resolution	low	low	any	any

The table above summaries the comparison between RGHT and other HT algorithms. Following are the time and space complexities of the mentioned algorithms.

Algorithm	**Time Complexity**	**Space Complexity**
HT	N/A	$\prod_{i=1}^{n} parameter_i$
GHT	$O(x_{size} \cdot y_{size} \cdot \#\theta \cdot \#x_{scale} \cdot \#y_{scale}$	$x_{size} \cdot y_{size}$
RHT	$O(\#\text{sample} \cdot f(n))$	#sample · cell size
RGHT	$O(\#\text{sample} \cdot c \cdot \#\theta)$	hash table size

Time complexity of classical HT is not available because HT is unable to detect arbitrary object of various scales and orientations. In RHT, $f(n)$ is a function which depends on the number of parameters n of the analytical object. For detecting line, Gaussian Elimination solves the linear system with $f(n) = n^3$. For detecting other analytical object, $f(n)$ will be larger. In RGHT, the constant term c is for solving the 4 equations (**Eq. 4** to **Eq. 7**) and $\#\theta$ denotes the number of iteration steps for detecting various rotated instances of the template object. If we only need to detect scaled objects without rotation, no iteration is required.

The main storage concern falls into the size of the accumulation structure. RGHT uses a 4-D hash table and the storage used is $10^4 \cdot p + s \cdot c$, where $10^4 = 10,000$ is the number of entries in the hash table, p is the size of a pointer for a list, s and c are the number of samples and cell size respectively. Since it is a table of lists, so each entry is simply

a pointer. The cell size c in RGHT is a constant, 5, since each cell only contains the tuple: (Hit rate, Ox, Oy, Sx, Sy)

3.5. Discussion and improvement

Image reduction is a technique for speeding up the algorithm. By shrinking the original image, less samples are needed to acquire certain accuracy. After locating the approximate position and scale of the targets, we can work on such area in more detail in the original image.

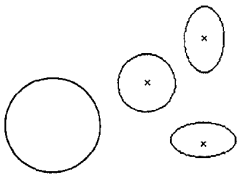
In addition, with the help of line extraction algorithms, such as Canny operator, we can remove some noise and unwanted disturbance in the target image. After preprocessing, we can have more accurate result. It is illustrated in our experiments.

4. Computer Experiments

Here are some experiments of using RGHT to detect various objects. All the experiments were conducted on a Sun SPARCstation 20 with SunOS Solaris 5.4 and the time were measured in CPU time.

Fig. 5 shows the result of detecting ellipses using the RGHT. The template image is a circle and the 3 targets are successfully marked with a cross at the estimated anchor point. The detection process costs 11 seconds with 200 iterations and the result is summarized in the table below.

$\hat{Ox}$	$\hat{Oy}$	$\hat{Sx}$	$\hat{Sy}$	Hit
37.5	81.2	0.3	0.3	30
93.0	39.5	0.2	0.5	12
91.7	140.5	0.4	0.2	8

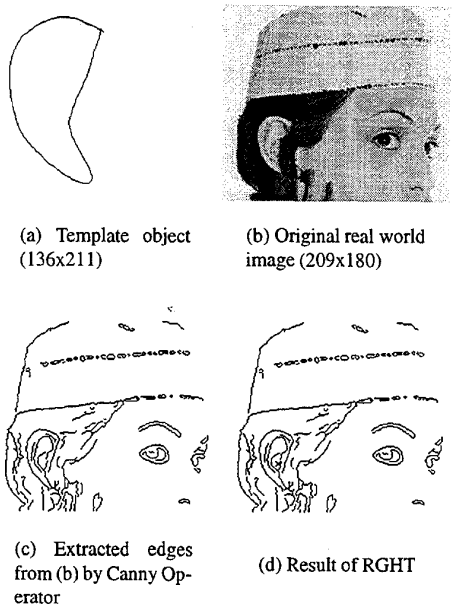


(a) (150x150) (b) (140x170)

Figure 5. Result of detecting ellipse

$\hat{Ox}$	$\hat{Oy}$	$\hat{Sx}$	$\hat{Sy}$	Hit
49.6	133.9	0.3	0.3	11

A human head is shown in **Fig. 6**. Here we use another arbitrary object as the template object. The table above summarizes the result. Note that the center of the ear is marked by a cross. The elapsed time is 10 seconds with 300 iterations.



(a) Template object (136x211)

(b) Original real world image (209x180)

(c) Extracted edges from (b) by Canny Operator

(d) Result of RGHT

Figure 6. Result of detecting object in real image

5. Conclusion

This paper proposes a new algorithm for object detection. The algorithm can work on grayscale images and detect arbitrary object with various scales and orientations. RGHT is actually a combination of GHT and RHT and it shares the advantages of the two algorithms. Since random sampling is used, there is a great save in time and space. Other advantages are high accuracy and arbitrary resolution.

References

[1] D. H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] Hough and P.V.C. Methods and Means for Recognizing Complex Patterns. U.S. Patent 3,069,654, 1962.

[3] L. Xu and E. Oja. Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities. *Image Understanding*, 57(2):131–154, 3 1993.

[4] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough Transform (RHT). *Pattern Recognition Letters*, 11:331–338, 5 1990.