Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT

PERGAMON

Pattern Recognition 37 (2004) 953–963

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

New memory- and computation-efficient hough transform for detecting lines

Kuo-Liang Chung[a,*,1], Teh-Chuan Chen[a], Wen-Ming Yan[b]

[a] *Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, ROC*

[b] *Department of Computer Science and Information Engineering, National Taiwan University, No. 1, Section 4, Roosevelt Road, Taipei, Taiwan 10764, ROC*

Received 23 May 2003; accepted 17 September 2003

Abstract

The slope-intercept Hough transform (SIHT) is one of the two types of line-detection methods. However, the disadvantage of the SIHT is its low memory utilization, say 50%. Based on the affine transformation, this paper presents a new method to improve the memory utilization of the SIHT from 50% to 100%. According to the proposed affine transformation, we first present a basic SIHT-based algorithm for detecting lines. Instead of concerning floating-point operations in the basic SIHT-based algorithm, an improved SIHT-based algorithm, which mainly concerns integer operations, is presented. Besides the memory utilization advantage, experimental results reveal that the improved SIHT-based algorithm has more than 60% execution time improvement ratio when compared to the basic SIHT-based algorithm and has more than 33% execution time improvement ratio when compared to another type of line-detection methods, such as the (r,θ)-based HT algorithm and its variant. The detailed complexity analyses for all the related algorithms are also investigated and we show that the time complexity required in the improved SIHT-based algorithm is the least.

Keywords: Affine transformation; Algorithms; Complexity; Hough transform; Line-detection; Parameter space; Slope-intercept space

1. Introduction

Detecting lines in a digital image is a classical problem in image processing field. Throughout this paper, we only focus on this line-detection issue since detecting the other shapes, such as circles and ellipses, is another research issue. Since Hough [1] presented the method for mapping the image domain to the slope-intercept (SI) parameter space to detect lines, the idea of the transformation from the image domain to the parameter space has become a popular technique [2]. Later, Wahl and Biland [3,4] proposed a twin SI approach with finite parameter space and the proposed method is called the SI Hough transform (SIHT). For deterministic HTs, besides the SIHT type, another well-known type is the (r,θ)-based HT proposed by Duda and Hart [5–8], where r and θ are the normal distance and normal angle of a line, respectively. For convenience, the (r,θ)-based HT is denoted by NHT. Note that the discussion of randomized HT algorithms for detecting lines is beyond the scope of this research and we only focus on the deterministic HT algorithms.

In Ref. [9], Risse indicated that the memory utilization of SIHT is 50%. This low memory utilization leads to the first motivation of this research. As the first motivation, this research wants to present a new approach to improve the memory utilization of the SIHT from 50% to 100%. Besides the first improvement, this paper further wants to present a new and faster SIHT-based algorithm which is also faster

* Corresponding author.
E-mail addresses: klchung@cs.ntust.edu.tw (K.-L. Chung), der@cc1.cyjcba.edu.tw (T.-C. Chen), ganboon@csie.ntu.edu.tw (W.-M. Yan).

[1] Supported by NSC89-2213-E011-061.

0031-3203/$30.00 © 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2003.09.008

than the NHT and its variant proposed by Ben-Tzvi and Sandler [10], called the CHT.

Based on the affine transformation, this paper presents a method to improve the memory utilization of the SIHT from 50% to 100%. According to the proposed affine transformation, we first present the first SIHT-based algorithm, denoted by FSIHT. Then, using the incremental approach, we improve the FSIHT such that only few multiplications are involved in the second algorithm called the SSIHT. Finally, by applying a sign testing technique, the third SIHT-based algorithm, denoted by TSIHT, is presented. In the TSIHT, it mainly concerns integer operations. In practice, the TSIHT can reduce the memory requirement of the SIHT [1] down to 67%. Some experiments have been carried out to compare the performance among the SIHT, FSIHT, SSIHT, TSIHT, NHT, and CHT. The experimental results reveal that the TSIHT is the fastest among the three proposed SIHT-based algorithms. Especially, the TSIHT has more than 60% execution time improvement ratio when compared to the FSIHT and has more than 33% execution time improvement ratio when compared to the NHT and CHT. The detailed complexity analyses for all the related algorithms are also investigated and we show that the time complexity required in the TSIHT is the least.

The remainder of this paper is organized as follows. Section 2 presents a set and geometrical approach to introduce the SI parameter space and analyzes the related memory utilization. Section 3 presents how to apply the affine transformation to map the SI parameter space into a fully utilized parameter space. On the newly transformed parameter space, Section 4 presents our novel algorithms. Section 5 illustrates the experimental results. Finally, some concluding remarks are addressed in Section 6.

2. The SI parameter space and memory utilization

In this section, the SI parameter space for detecting lines is defined explained first. Because we are dealing with digital images, the corresponding curves of the collinear points in digital image are hard to exactly meet at one point in the parameter space. Instead, these curves will pass through a region, i.e. cell, in the parameter space. Therefore, for implementation, we uniformly quantize the parameter space into many small cells and use voting along curves on these cells. When we complete the voting process, we report those lines corresponding to those cells whose counts exceed a predetermined threshold. Because we only concern with lines existing in the digitized image, many cells in the quantized parameter space may never be used. That is, no vote will be occurred in those useless cells. For example, the cell with slope 0 and intercept -1 is useless. Let the number of cells in the parameter space be R and the number of useless cells be U, then the memory utilization is defined as $(R-U)/R$. Low memory utilization will lead to waste the memory. By using the set and geometrical viewpoint, in the remainder of this section, we provide a new proof to show that the memory utilization of the SIHT is 50% as shown in Ref. [9].

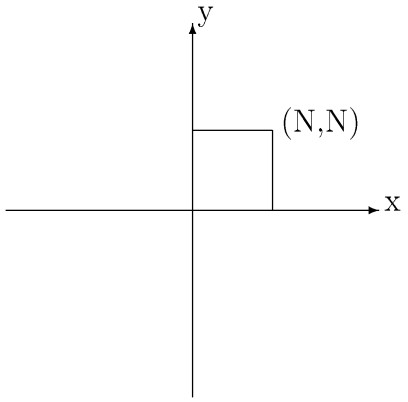

Fig. 1. The image domain I.

The basic concept of the SIHT is that a line passing through a point (x, y) in the image domain with slope $|a| \leqslant 1$ ($|a'| > 1$) and intercept b can be represented by equation $y = ax + b$ ($x = y/|a'| + b$). Using this equation, we can map the point (x, y) in xy (yx) space, i.e. the image domain, to a dual curve in the SI parameter space. Further, an intersection between two curves for the two points corresponds to a line which passes through these two points. That is, the mapping has the property that collinear points in xy space will meet in the intersection of the corresponding dual curves in the SI parameter space.

Suppose we are given an image domain I as shown in Fig. 1 for $0 \leqslant x \leqslant N$ and $0 \leqslant y \leqslant N$.

For each point (x, y) in I, let a and b be the slope parameter and the intercept parameter of a straight line, respectively. Then every line passing through it can be represented by the SI form $y = ax + b$ in xy space or $x = ay + b$ in yx space (exchange x and y) with $-1 \leqslant a \leqslant 1$. The restriction $-1 \leqslant a \leqslant 1$ on the slope a is reasonable since we can change the line equation $y = ax + b$ with $|a| > 1$ to the form $x = ay + b$ with $|a| < 1$ and view the line with absolute value of slope less than 1 in yx space. Without losing generality, we only discuss the case $y = ax + b$ with slope $|a| \leqslant 1$ throughout this paper. If we rewrite the equation $y = ax + b$ to $b = -ax + y$ and fix a point (x, y), we can map that point (x, y) to a dual line in the SI parameter space. Furthermore, all collinear points along a straight line in xy space will have dual lines in the SI parameter space such that these dual lines intersect at the same point.

In order to investigate the possible range of the SI parameter space, we define a dual representation to map that point in xy space to a line segment in the SI parameter space:

$$H(x, y) = \{(a, b) : -1 \leqslant a \leqslant 1, b = -ax + y\}.$$

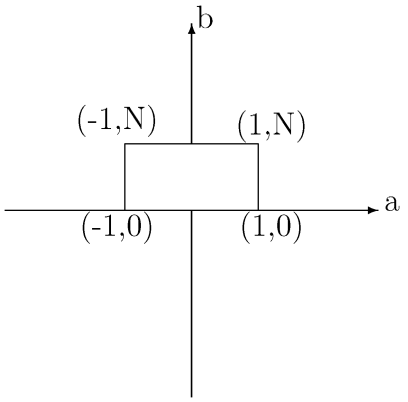

Fig. 2. The region for $\bigcup_{0\leqslant y\leqslant N} H(0,y)$.

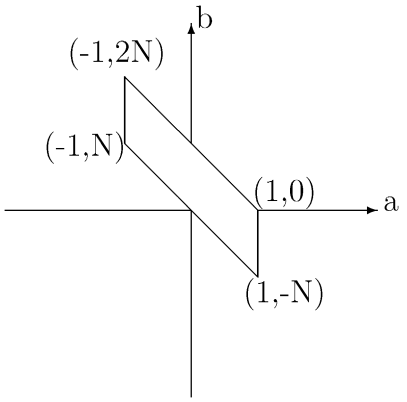

Fig. 3. The region for $\bigcup_{0\leqslant y\leqslant N} H(N,y)$.

Besides, let us define two special points in the SI parameter space

$$P_1^{x,y}=(-1,x+y) \quad \text{and} \quad P_2^{x,y}=(1,-x+y),$$

then for each point (x,y) in the image domain I, its dual representation for $H(x,y)$ can be represented by the line segment $\overline{P_1^{x,y}P_2^{x,y}}$ which is connected by the two points $(-1,x+y)$ and $(1,-x+y)$ in the SI parameter space.

Following the definition of $H(x,y)$, we now define a rectangular region in the SI parameter space. From $H(0,y)=\overline{P_1^{0,y}P_2^{0,y}}$, where $P_1^{0,y}=(-1,y)$ and $P_2^{0,y}=(1,y)$,

$$\bigcup_{0\leqslant y\leqslant N} H(0,y)$$

represents a rectangular region as shown in Fig. 2. Similarly, from $H(N,y)=\overline{P_1^{N,y}P_2^{N,y}}$, where $P_1^{N,y}=(-1,N+y)$ and $P_2^{0,y}=(1,-N+y)$,

$$\bigcup_{0\leqslant y\leqslant N} H(N,y)$$

represents a parallelogram as shown in Fig. 3.

Let P, as shown in Fig. 4, denote the set of the union of the regions in Figs. 2 and 3. It is easy to know that

$$P=\left[\bigcup_{0\leqslant y\leqslant N} H(0,y)\right]\bigcup\left[\bigcup_{0\leqslant y\leqslant N} H(N,y)\right]$$

$$\subseteq \bigcup_{(x,y)\text{ in }I} H(x,y).$$

Moreover, for each point (x,y) in I, since $0\leqslant x+y\leqslant 2N$ and $-N\leqslant -x+y\leqslant N$, we have the two points $P_1^{x,y}=(-1,x+y)$ and $P_2^{x,y}=(1,-x+y)$ which are always in P. In addition, the midpoint $(0,y)$ of $\overline{P_1^{x,y}P_2^{x,y}}$ is also in P. So, we have

$$\bigcup_{(x,y)\text{ in }I} H(x,y)\subseteq P.$$

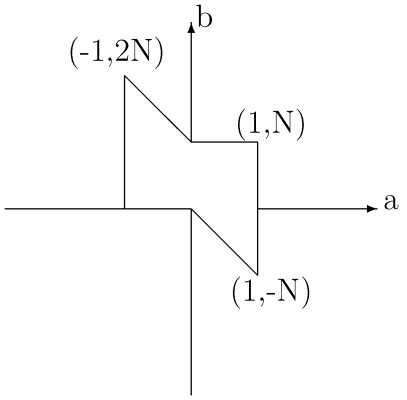

Fig. 4. The region P.

Consequently, we obtain that

$$\bigcup_{(x,y)\text{ in }I} H(x,y)=P.$$

Therefore, the region of the SI parameter space before quantization is P whose area, i.e. size, is $3N$.

To implement the SIHT, we usually evenly quantize the SI parameter space into a form of rectangular tessellation with respect to a 2D memory. Here, the smallest rectangular region of the SI parameter space which contains P is $[-1,1]\times[-N,2N]$. It is easy to observe that two areas in that rectangular region will never be used.

Let the size of the quantized SI parameter space be the number of cells required in the rectangular tessellation. We have the following result.

Lemma 1. *The size of the smallest rectangular region of the quantized SI parameter space, i.e. the memory, is* $6N$, *and the memory utilization of SIHT is* 50% (*i.e.* $(6N-3N)/6N$).

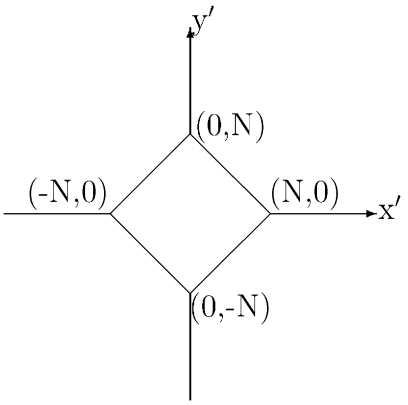

Fig. 5. The image domain I'.

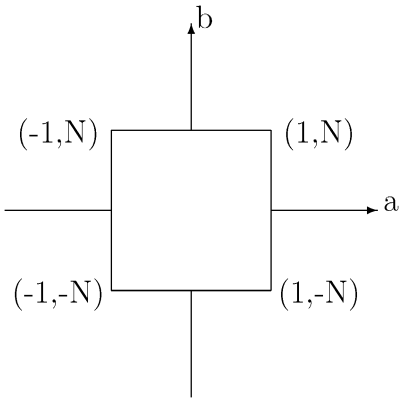

Fig. 6. The region P'.

3. The proposed affine transformation

In this section, we present an affine transformation-based approach to reach 100% memory utilization and reduce the memory requirement in the quantized SI parameter space from $6N$ to $4N$ (see Lemma 1).

Our affine transformation is given by

$$x' = x - y,$$
$$y' = x + y - N, \tag{1}$$

where (x, y) is a point in the image domain I. After applying our affine transformation, we can map the region of the image domain I as shown in Fig. 1 into the region of Fig. 5, where the image domain of Fig. 5 is denoted by I'. It is not hard to see that the transformed image domain I' in $x'y'$ space can be represented as the equation $|x'| + |y'| \leqslant N$.

Following the same discussion as in Section 2, we map each point (x', y') in I' to its dual segment $H(x', y')$ in the SI parameter space. Here, $H(x', y')$ can be represented by the line segment which is connected by two points $(-1, x' + y')$ and $(1, -x' + y')$. Note that whenever (x', y') in I', (x', y') satisfies the constraint $|\pm x' + y'| \leqslant |x'| + |y'| \leqslant N$.

Let $P' = \{(a, b) : -1 \leqslant a \leqslant 1, -N \leqslant b \leqslant N\}$ as shown in Fig. 6, then we have $H(x', y') \subseteq P'$ whenever (x', y') in I'. It yields

$$\bigcup_{(x',y') \text{ in } I'} H(x', y') \subseteq P'.$$

In addition, for $|y'| \leqslant N$, $(0, y')$ is within the domain I' and $H(0, y')$ can be represented by the line segment which is connected by two points $(-1, y')$ and $(1, y')$. From the definition of P', we know that

$$P' = \bigcup_{|y'| \leqslant N} H(0, y').$$

Because of

$$P' = \bigcup_{|y'| \leqslant N} H(0, y') \subseteq \bigcup_{(x',y') \text{ in } I'} H(x', y'),$$

we have

$$\bigcup_{(x',y') \text{ in } I'} H(x', y') = P'.$$

Using the above affine transformation, we actually have a more compact memory in the newly transformed parameter space. From Fig. 6, since the region P' is rectangular and no area in P' is useless, we have the following result.

Lemma 2. *The size of the smallest rectangular region in the newly transformed of the parameter space is* $4N$, *and the memory utilization is* 100%.

Combining the results in Lemmas 1 and 2, we conclude that the memory saving ratio of our proposed parameter space over the previous SI parameter space is $(6N - 4N)/(6N) = 1/3$.

From Eq. (1), there is a one-to-one correspondence between I and I'. The corresponding relation between them can be obtained by the substitution technique. We have the following two results.

Theorem 3. *If there is a line* $y' = ax' + b$ *in* I' *for* $|a| \leqslant 1$, *then there is a line* $x = (b + N)/2$ *in* I *when* $a = -1$; *otherwise, there is a line* $y = (a - 1)/(a + 1)x + (b + N)/(a + 1)$ *in* I.

Proof. From Eq. (1), the line $y' = ax' + b$ in I' corresponds to the line $(x + y - N) = a(x - y) + b$, which is equal to $(a + 1)y = (a - 1)x + b + N$, in I. Considering $a = -1$, we have a line $x = (b + N)/2$ in I. Otherwise, it is easy to derive a line $y = (a - 1)/(a + 1)x + (b + N)/(a + 1)$ in I.

Theorem 4. *If there is a line $x' = ay' + b$ in I' for $|a| \leqslant 1$, then there is a line $y = (N - b)/2$ in I when $a = 1$; otherwise, there is a line $x = (1+a)/(1-a)y + (b - aN)/(1-a)$ in I.*

Proof. From Eq. (1), the line $x' = ay' + b$ in I' corresponds to the line $(x - y) = a(x + y - N) + b$, which is equal to $(1 - a)x = (1 + a)y + b - aN$, in I. Considering $a = 1$, we have a line $y = (N - b)/2$ in I. Otherwise, it is easy to derive a line $x = (1 + a)/(1 - a)y + (b - aN)/(1 - a)$ in I.

4. The proposed algorithms

In this section, based on the above affine transformation, we first present a basic SIHT algorithm, called the FSIHT, which involves a large amount of multiplications and floating-point (FL) operations. Note that the difference between the SIHT [3,4] and the proposed FSIHT is that the latter employs the affine transformation to increase the memory utilization from 50% to 100%. However, we will give the detailed time complexity analyses for both the SIHT and the FSIHT.

Next, we present the second algorithm, which is an improved version of the FSIHT, and the proposed algorithm, SSIHT, involves few multiplications. Finally, we present the third algorithm, TSIHT, which mainly concerns integer operations and is the fastest when compared with the FSIHT and SSIHT. Even compared with SIHT, NHT, and CHT, the proposed TSIHT is still the fastest.

4.1. The first algorithm: FSIHT

For implementation, we first translate the rectangular region of Fig. 6 to the first quadrant. So, we let

$$\hat{H}(x', y') = H(x', y') + (1, N) = \{(a, b) : 0 \leqslant a \leqslant 2, \quad b = -(a - 1)x' + y' + N\}$$

and

$$\hat{P} = P' + (1, N) = \{(a, b) : 0 \leqslant a \leqslant 2, 0 \leqslant b \leqslant 2N\}.$$

Using the similar affine transformation technique, it can be verified that

$$\bigcup_{(x', y') \text{ in } I'} \hat{H}(x', y') = \hat{P}.$$

For implementation, we evenly quantize the slope interval $[0, 2]$ into $2N + 1$ quantized values, i.e. $0, 1/N, 2/N, \ldots,$ and 2. Furthermore, we evenly quantize the intercept interval $[0, 2N]$ into $2N + 1$ integers, i.e. $0, 1, 2, \ldots,$ and $2N$. Then the proposed first algorithm is presented as follows:

ALGORITHM: FSIHT

```
{initially, the 2D array, namely HS, for parameter space is
  set to zero}
  HS ← [0]
  {perform affine transformation for each edge pixel (x, y)
    in the image domain I}
  {this is the main difference between the SIHT and the
    FSIHT}
for each edge pixel (x, y) in I
  x' ← x − y
  y' ← x + y − N
  for i ← 0 to 2N {voting process}
    b ← int((1 − i/N)x' + y' + N) {deduce from the
      equation y' = (i/N − 1)x' + (b − N)}
    HS[i, b] ← HS[i, b] + 1 {voting operation: count the
      number of collinear edge pixels}
  endfor
endfor
```

Given a threshold value T, suppose the value of $HS[i, b]$ is greater than T, we thus detect a line, $y' = (i/N - 1)x' + (b - N)$, in the domain I'. That is, we detect a line, $b = (1 - i/N)(x - y) + x + y$, in the domain I.

Assume the cost of one subtraction (division) is equal to one addition (multiplication). For each edge pixel (x, y), the FSIHT needs three integer additions for doing our affine transformation. Besides, it needs $2N + 1$ voting operations, $2N + 1$ operations for converting FL values to integer values (see 'int' in the FSIHT algorithm), $3(2N + 1)$ FL additions, and $2(2N + 1)$ FL multiplications. For simplicity, let INT be one operation for converting one FL value to the integer value. Here, the voting operation just needs one integer addition. Combining the time complexity required for $|a| \leqslant 1$ and $|a| > 1$, the FSIHT needs $3 + 2(2N + 1)$ integer additions, $6(2N + 1)$ FL additions, $4(2N + 1)$ FL multiplications, and $2(2N + 1)$ INTs for each edge pixel.

As for the SIHT, it does not need three integer additions for doing our affine transformation for each edge pixel. Moreover, the statement $b \leftarrow \text{int}((1 - i/N)x' + y' + N)$ should be replaced by $b \leftarrow \text{int}((1 - i/N)x + y + N)$. So it needs $2(2N + 1)$ integer additions, $6(2N + 1)$ FL additions, $4(2N + 1)$ FL multiplications, and $2(2N + 1)$ INTs. However, in SIHT, it needs $(3N + 1) \times (2N + 1)$ array memory. Therefore, SIHT needs more time to set the array to zero for initialization and to check whether each cell in the array is greater than the threshold T or not. Specifically, the array memory required in the proposed algorithm FSIHT is of size $(2N + 1) \times (2N + 1)$ and it leads to save some amount of memory-checking time.

From the above discussion, we can see that the time complexities of the SIHT and the FSIHT do not make much difference. But, the proposed FSIHT reduces the memory requirement of the SIHT down to 67%.

4.2. The second algorithm: SSIHT

Since one multiplication costs some additions, we now give an improved algorithm of the FSIHT, say SSIHT, which only involves few multiplications during the voting process. The improved algorithm SSIHT is motivated from the fact that we can change the expression of the equation $b = (1 - i/N)x' + y' + N$ into $b = x' + y' + N - i(x'/N)$. So, for giving (x', y'), the next voting value b for $i + 1$ can be obtained using the incremental approach with the incremental value $-x'/N$ each time. The formal SSIHT algorithm is listed below.

ALGORITHM: SSIHT

```
{initially, the 2D array HS for parameter space is set to zero}
  HS ← [0]
for each edge pixel (x, y) in I
  x' ← x − y
  y' ← x + y − N
  s ← −x'/N {s denotes the incremental value}
  r ← x' + y' + N − s {r is the real intercept; here we set
    the initial value of r}
  for i ← 0 to 2N {voting process}
    r ← r + s {incremental step}
    b ← int(r) {find b who is the closest integer value
      of r}
    HS[i, b] ← HS[i, b] + 1 {count the number of collinear
edge pixels}
  endfor
    endfor
```

Considering the two cases, $|a| \leqslant 1$ and $|a| > 1$, for each edge pixel, the SSIHT algorithm needs $3 + 2(2N + 1)$ integer additions for doing our affine transformation and voting operations. Six FL additions and two FL multiplications are needed for preparing some initial values. In addition, it needs extra $2(2N + 1)$ FL additions and $2(2N + 1)$ INTs.

4.3. The third algorithm: TSIHT

In the above SSIHT algorithm, it is observed that the incremental value $-x'/N$ is not integer and has absolute value less than or equal to 1. That is, the next voting value b is either incremented by ± 1 (depends on the sign of the incremental value) or equals to the previous voting value. In what follows, we present an improved algorithm of the SSIHT, say TSIHT, which mainly concerns integer operations. Basically, we use a discriminator to decide whether the next voting is incremented by ± 1 or not. The proposed TSIHT algorithm is listed below.

ALGORITHM: TSIHT

```
{initially, the 2D array HS for parameter space is set to zero}
  HS ← [0]
for each edge pixel (x, y) in I
  x' ← x − y
  y' ← x + y − N
  b ← x' + y' + N
  HS[0, b] ← HS[0, b] + 1 {voting for the case when slope
    a = 0}
  if x' > 0 then {b will be nonincreasing}
    d ← ⌈N/2⌉ {discriminator}
    for i ← 1 to 2N
      d ← d − x' {next discriminator}
      d ⩽ 0 then {subtract 1 from b}
        b ← b − 1
        d ← d + N
      end if
      HS[i, b] ← HS[i, b] + 1 {count the number of
        collinear edge pixels}
    end for
  else if x' < 0 then {b will be nondecreasing}
    d ← −⌈N/2⌉ {discriminator}
    for i ← 1 to 2N
      d ← d − x'
      if d ⩾ 0 then {add 1 to b}
        b ← b + 1
        d ← d − N
      end if
      HS[i, b] ← HS[i, b] + 1 {count the number of
        collinear edge pixels}
    end for
  else {if x' = 0, then b will be a constant}
    for i ← 1 to 2N
      HS[i, b] ← HS[i, b] + 1 {count the number of
        collinear edge pixels}
    end for
  end if
endfor
```

Considering the two cases, $|a| \leqslant 1$ and $|a| > 1$, for each edge pixel, the TSIHT algorithm needs $5 + 2(2N + 1)$ integer additions for performing our affine transformation, setting the initial value b, and voting operations. In addition, two ceiling operations are needed. Furthermore, during the voting process, it only needs $12N$ integer additions and $2(2 + 2N)$ sign testing operations in the worst case.

4.4. Complexity analysis for NHT

As mentioned in Section 1, we hope to include the (r, θ)-based HT proposed by Duda and Hart [6], denoted by NHT, for comparison with the FSIHT, SSIHT, and TSIHT. Since the time complexity of the NHT depends on the dimension of quantization of the parameter, for fairness, we

Table 1
Memory and time complexity comparison among NHT, SIHT, FSIHT, SSIHT, and TSIHT

	NHT	SIHT	FSIHT	SSIHT	TSIHT
cos	$2(2N+1)$	0	0	0	0
$F_\times$	$4(2N+1)+6$	$4(2N+1)$	$4(2N+1)$	2	0
F_+	$2(2N+1)+1$	$6(2N+1)$	$6(2N+1)$	$6+2(2N+1)$	0
INT	$(2N+1)$	$2(2N+1)$	$2(2N+1)$	$2(2N+1)$	0
I_+	$(2N+1)$	$2(2N+1)$	$3+2(2N+1)$	$3+2(2N+1)$	$7+16N$
Ceiling	0	0	0	0	2
Sign	0	0	0	0	$2(2+2N)$
Size	$(2N+1)$ $\times(2N+1)$	$(3N+1)$ $\times(2N+1)$	$(2N+1)$ $\times(2N+1)$	$(2N+1)$ $\times(2N+1)$	$(2N+1)$ $\times(2N+1)$

arrange the NHT with the same the dimension of the parameter space quantization as in the above three proposed algorithms.

We first outline the NHT. The NHT using the normal angle θ and normal distance r, where $r = x\cos\theta + y\sin\theta$, to represent a line and the parameter space can be restricted to $[0,\pi]\times[-N,\sqrt{2}N]$. For each edge pixel (x, y), during the ith voting, $i = 0,\ldots,2N$, it needs to calculate $\theta \leftarrow i(\Delta\theta)$, $r \leftarrow x\cos\theta + y\sin\theta, b \leftarrow \text{int}((r+N)/(\Delta r))$, and one voting operation. Here, $\Delta\theta = \pi/(2N)$ and $\Delta r = (\sqrt{2}N+N)/(2N)$ are the quantization interval for θ and r, respectively, and can be calculated in the initialization phase; b is used to determine the index number of the r-axis in the quantized (r,θ) parameter space. Assume the cost of one $\cos\theta$ operation is equal to one $\sin\theta$ operation and the cost of one square root operation is equal to one FL multiplication. Totally, for each edge pixel, the NHT needs $2(2N+1)$cos operations, $4(2N+1)+6$ FL multiplications, $2(2N+1)+1$ FL additions, $(2N+1)$ INTs, and $(2N+1)$ integer additions.

4.5. Memory and time comparison among NHT, SIHT, FSIHT, SSIHT, and TSIHT

The size of the memory array required in the parameter space and the time complexity needed in the NHT, SIHT, FSIHT, SSIHT, and TSIHT, respectively, for each edge pixel are summarized in Table 1. In Table 1, the meaning of cos, $F_\times$, F_+, INT, I_+, ceiling, sign, size, and cost are explained as follows: cos is the number of cos operations; $F_\times$ is the number of FL multiplications; F_+ is the number of FL additions; INT is the number of operations for converting FL values to integers; I_+ is the number of integer additions; Ceiling is the number of ceiling operations; Sign is the number of sign testing operations and Size is the size of memory array needed in the parameter space.

From Table 1, it is observed that the time complexity required in the proposed FSIHT is nearly the same as that in the SIHT. The main reason is that the memory size used in the FSIHT is less than that in the SIHT although the FSIHT takes three integer additions to perform the affine transformation on each edge pixel. Unlike the other algorithms, such as NHT, SIHT, FSIHT, and SSIHT, the proposed TSIHT does not involve any FL operations and in practice, the TSIHT will be the fastest. The experiments in the next section will confirm this.

5. Experimental results

In order to demonstrate the thorough time comparison, besides the above five algorithms, the comparison also covers the CHT, the SIHT′, and the SIHT″, where SIHT′ (SIHT″) denotes the SIHT but employing the incremental approach (sign testing technique) which is used in the SSIHT (TSIHT). In implementation, three real images as shown in Fig. 7 are used to test the time performance, each image with size 256×256. All the experiments are performed on a Pentium III 733 MHz computer using C language.

Figs. 7(a)–(c) are the road-image, the airport-image, and floor-image, respectively. After applying the edge detection method using Laplacian operator [11], Fig. 8 shows the three resulting images and the number of edge pixels are 1090, 1780, and 3207, respectively. The detected lines using our proposed affine transformation-based algorithms are shown in Fig. 9. Each line in Fig. 9 is drawn using the corresponding two parameters of that line detected in the algorithms.

From the equation $r = x\cos\theta + y\sin\theta$ used in the NHT, it is easy to know that the incremental approach and the sign testing technique mentioned previously cannot be applied to improve the NHT. Table 2 shows the time comparison among SIHT, SIHT′, TSIHT″, FSIHT, SSIHT, and TSIHT. In Table 2, the second-to-fourth columns denote the names of line-detection algorithms and the execution time required in terms of milliseconds. The notations $IR^{\text{SIHT}}_{\text{SIHT}'}$, $IR^{\text{SIHT}}_{\text{SIHT}''}$, $IR^{\text{FSIHT}}_{\text{SSIHT}}$, and $(IR^{\text{FSIHT}}_{rmTSIHT})$ denote the time

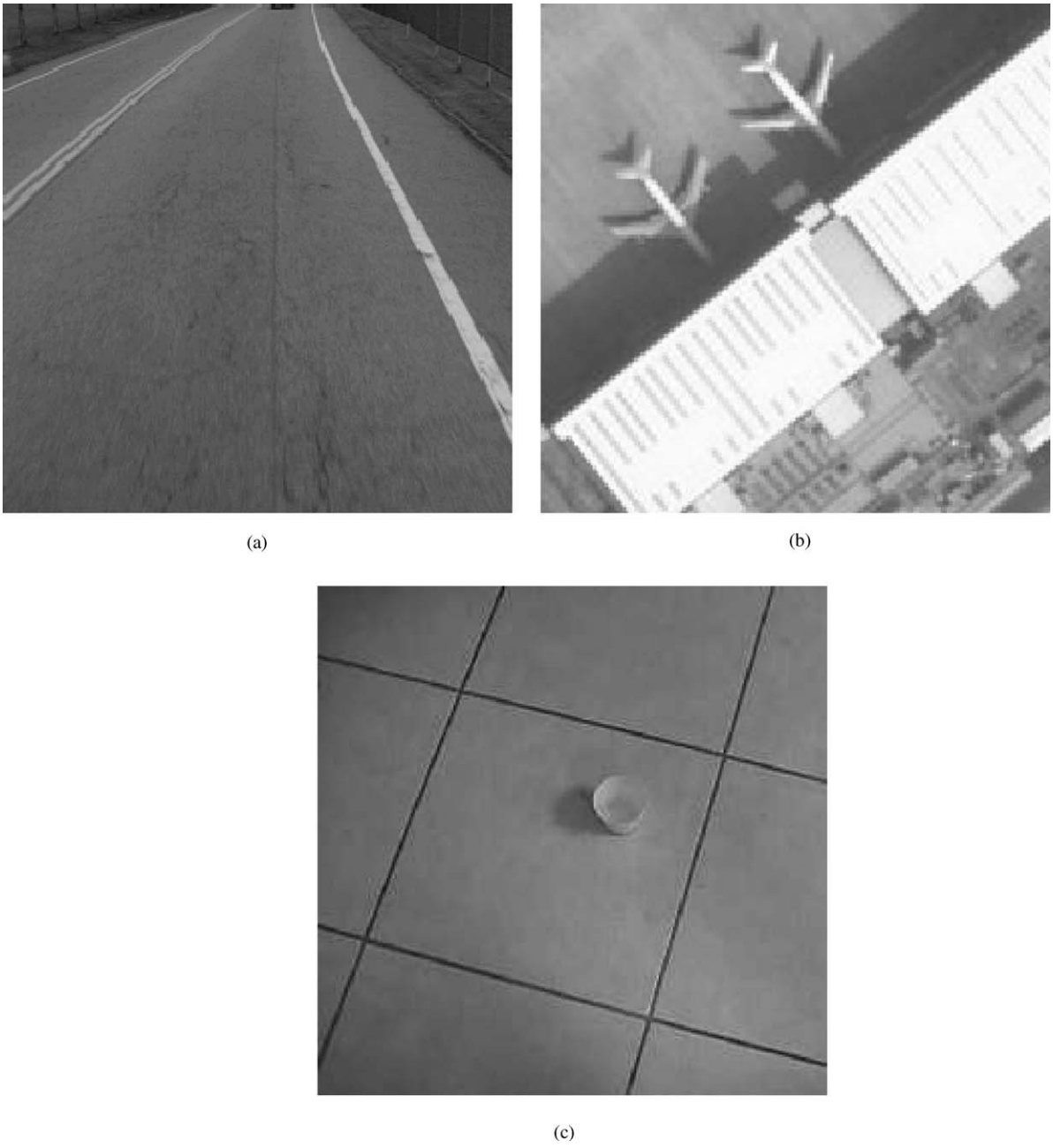

Fig. 7. Three testing images. (a) Road-image. (b) Airport-image. (c) Floor-image.

improvement ratios (SIHT−SIHT′)/SIHT, (SIHT−SIHT″)/SIHT, (FSIHT−SSIHT)/FSIHT, and (FSIHT−TSIHT)/FSIHT, respectively.

From Table 2, it is observed that the SIHT″ (TSIHT) algorithm has more than 60% execution time improvement ratio when compared to the SIHT (FSIHT) algorithm. It is implied that the incremental approach and the sign testing technique really can improve the original SIHT algorithm significantly. Furthermore, let us compare the time between SIHT″ and TSIHT. Table 3 shows that the execution time improvement ratio of the TSIHT over the SIHT″ is more than 0.02%. Besides the computational advantage, the proposed TSIHT algorithm reduces the memory requirement of the SIHT″ down to 67% (see Lemmas 1 and 2).

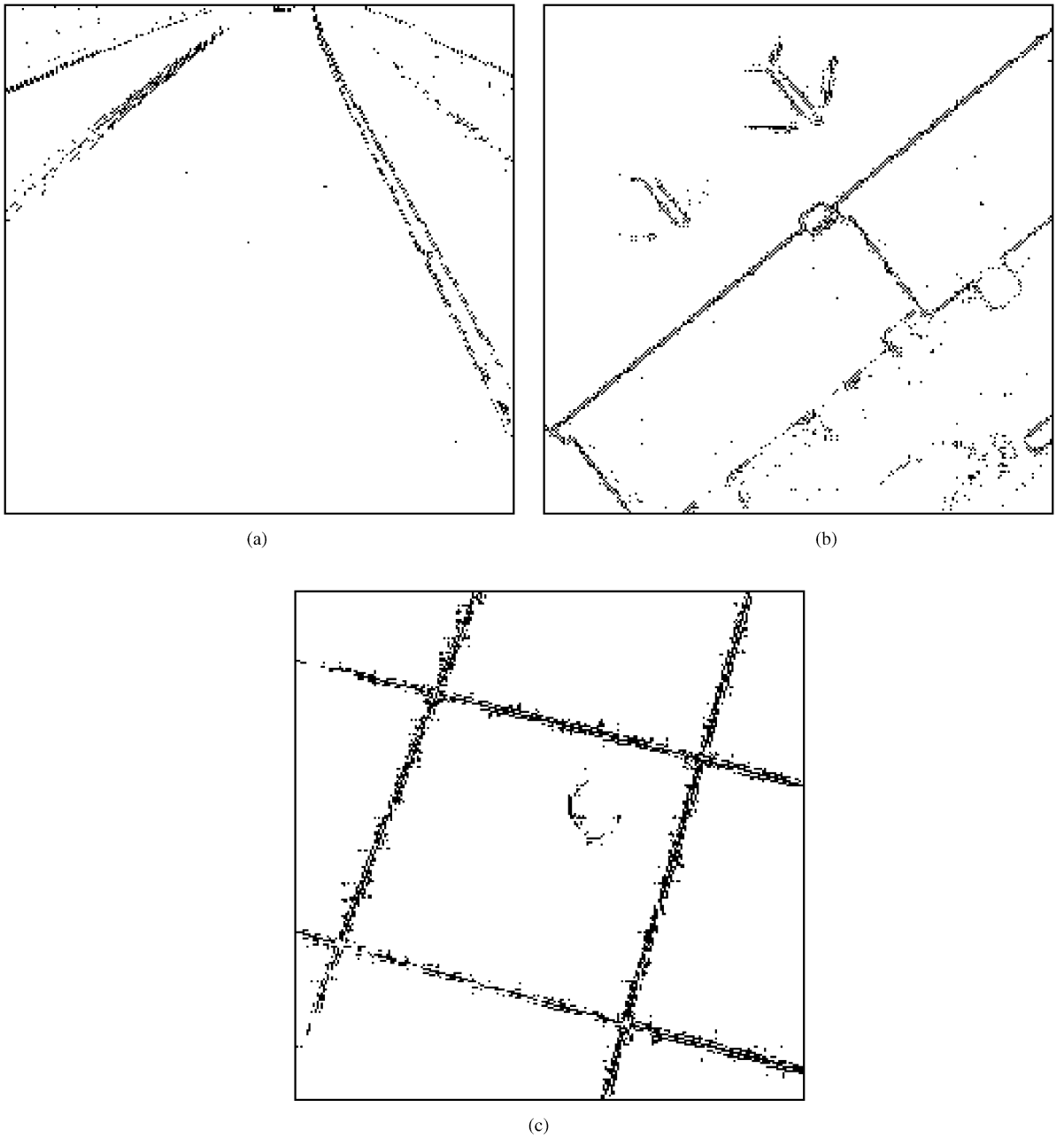

Fig. 8. The three sets of edge pixels for Fig. 7. (a) Edge pixels for road-image. (b) Edge pixels for airport-image. (c) Edge pixels for floor-image.

Let us further compare the time comparison among the TSIHT, NHT, CHT, and NHTT, where the NHTT is the NHT in which the sine and cosine are implemented by using the lookup table. Note that the sine and cosine in the NHT are implemented by calling the subroutines, sin( ) and cos( ). Table 4 shows that the proposed TSIHT algorithm has more than 33% execution time improvement ratio when compared to the NHT, NHTT, and CHT.

From Tables 2–4, we conclude that besides the memory-saving advantage, the proposed TSIHT is the fastest among the SIHT, SIHT′, SIHT″, FSIHT, SSIHT, TSIHT, NHT, NHTT, and CHT.

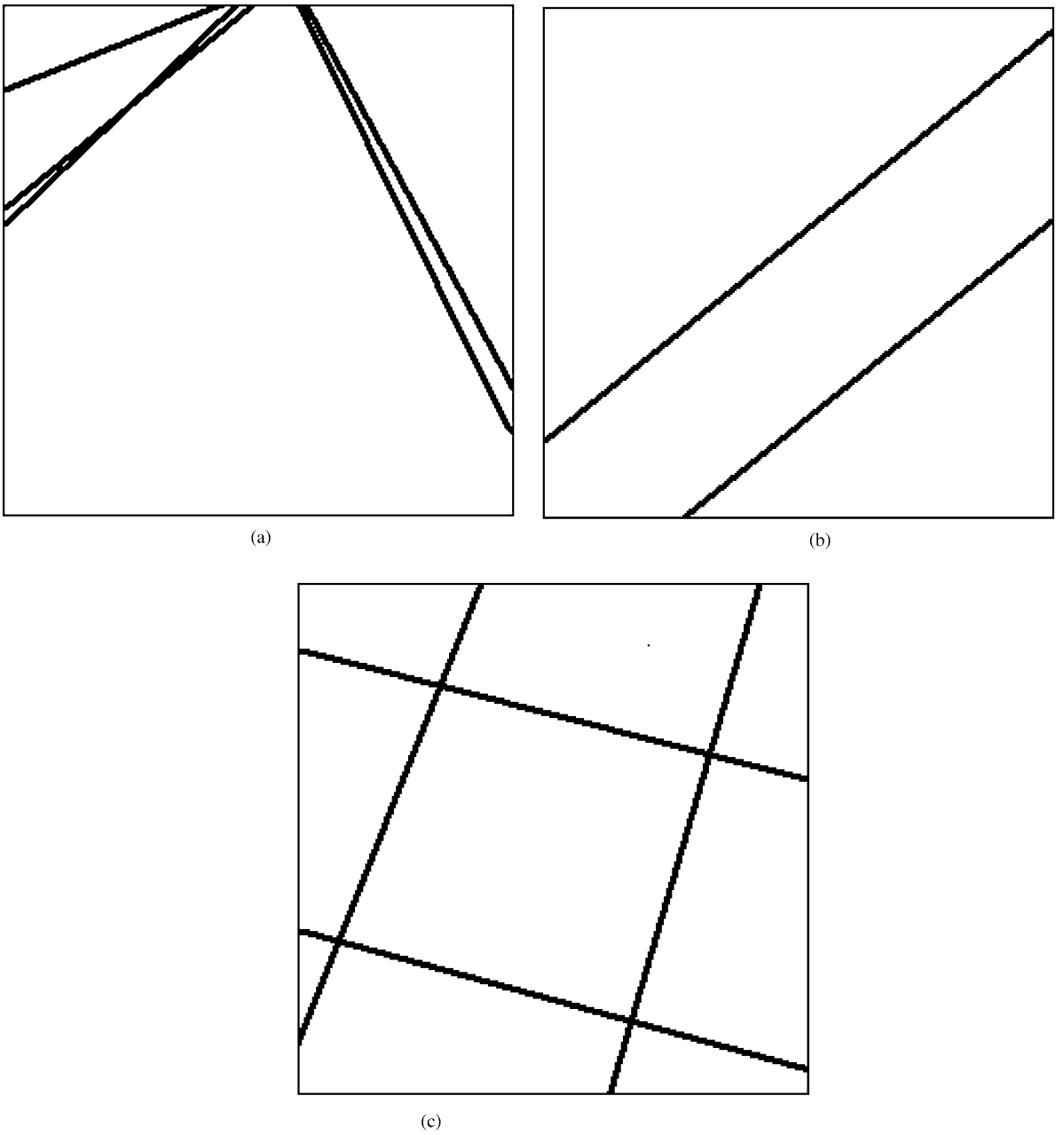

Fig. 9. The resulting detected lines. (a) Detected lines in road-image. (b) Detected lines in airport-image. (c) Detected lines in floor-image.

6. Conclusions

This paper has presented an affine transformation-based approach to improve the memory utilization of the SIHT [3,4] from 50% to 100%. We also provide a detailed proof. According to the proposed affine transformation, we present three SIHT-based algorithms to improve the original SIHT. Among these three proposed algorithms and the SIHT, the TSIHT is the fastest. In the TSIHT, it mainly concerns integer operations. The detailed time complexity for each algorithm is also investigated. In fact, the incremental approach and the sign testing technique used in the proposed TSIHT can also be applied to speed up the SIHT. In addition, the NHT, the NHTT, and the CHT are covered to test the performance with the proposed TSIHT.

Table 2
Time comparison among SIHT, SIHT′, SIHT″, FSIHT, SSIHT, and TSIHT

Image	SIHT	SIHT′	SIHT″	$IR^{\text{SIHT}}_{\text{SIHT}'}$	$IR^{\text{SIHT}}_{\text{SIHT}''}$
Road	256	232	96	0.09	0.63
Airport	410	372	163	0.09	0.60
Floor	778	606	220	0.22	0.72
	FSIHT	SSIHT	TSIHT	$IR^{\text{FSIHT}}_{\text{SSIHT}}$	$IR^{\text{FSIHT}}_{\text{TSIHT}}$
Road	253	225	91	0.11	0.64
Airport	415	366	155	0.12	0.63
Floor	793	602	215	0.24	0.73

Table 3
Time comparison between SIHT″ and TSIHT

Image	SIHT″	TSIHT	$IR^{\text{SIHT}''}_{\text{TSIHT}}$
Road	96	91	0.05
Airport	163	155	0.05
Floor	220	215	0.02

Table 4
Time comparison among NHT, NHTT, CHT, and the proposed TSIHT

Image	NHT	NHTT	CHT	TSIHT	$IR^{\text{NHT}}_{\text{TSIHT}}$	$IR^{\text{NHTT}}_{\text{TSIHT}}$	$IR^{\text{CHT}}_{\text{TSIHT}}$
Road	682	144	147	91	0.87	0.37	0.38
Airport	1110	234	230	155	0.86	0.34	0.33
Floor	1980	404	330	215	0.89	0.47	0.35

Some experiments have been carried out to compare the performance among the SIHT, SIHT′, SIHT″, FSIHT, SSIHT, TSIHT, NHT, NHTT, and CHT. The experimental results reveal that the TSIHT is the fastest among the concerning nine algorithms. Especially, the TSIHT has more than 60% execution time improvement ratio when compared to the FSIHT and has more than 33% execution time improvement ratio when compared to the NHT and CHT.

References

[1] P.V.C. Hough, Method and means for recognizing complex patterns, U.S. Patent 3,069,654, December 18, 1962.
[2] V.F. Leavers, Survey: which hough transform, CVGIP: Image Understanding 58 (2) (1993) 250–264.
[3] H.P. Biland, The recognition and volumetric description of three-dimensional polyhedral scenes in analysis of Hough-space structures, Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, ETH Zürich, 1987.
[4] F.M. Wahl, H.P. Biland, Decomposition of polyhedral scenes in Hough space, in: 18th International Conference on Pattern Recognition, IEEE, New York, 1986, pp. 78–84.
[5] D.H. Ballard, C.M. Brown, Computer Vision, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[6] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.
[7] A. Rosenfeld, Picture Processing by Computer, Academic Press, New York, 1969.
[8] A. Rosenfeld, A.C. Kak, Digital Picture Processing, 2nd Edition, Academic Press, San Diego, CA, 1986.
[9] T. Risse, Hough transform for line recognition: complexity and evidence accumulation and cluster detection, Comput. Vision Graphics Image Process. 46 (1989) 327–345.
[10] D. Ben-Tzvi, M.B. Sandler, A combinatorial hough transform, Pattern Recognition Lett. 11 (1990) 167–174.
[11] R.C. Gonzalez, R.E. Woods, Digital Image Processing. Addison-Wesley, New York, 1992, pp. 435–438.

About the Author—KUO-LIANG CHUNG received the BS, MS, and Ph.D. degrees in Computer Science and Information Engineering from National Taiwan University in 1982, 1984, and 1990, respectively. From 1984 to 1986, he was a soldier. From 1986 to 1987, he was a research assistant in the Institute of Information Science, Academic Sinica. He has been a Professor in the Department of Computer Science and Information Engineering at National Taiwan University of Science and Technology since 1995. Now he is the Chairman. Prof. Chung received the Distinguished Professor Award from the Chinese Institute of Engineers in May 2001. He is also an IEEE senior member. His research interests include image compression, image processing, video compression, coding theory, and algorithms.

About the Author—TEH-CHUAN CHEN received the MS degree in Mathematics from National Tsing-Hua University. Dr. Chen received the Ph.D. degree from National Taiwan University of Science and Technology. His research interests include image processing, pattern recognition, and algorithms.

About the Author—WEN-MING YAN received the BS and MS degrees in Mathematics from National Taiwan University. He is an Associate Professor in the Department of Computer Science and Information Engineering at National Taiwan University. Prof. Yan's research interests include image processing, coding theory, and algorithms.