Pergamon

Pattern Recognition, Vol. 28, No. 12, pp. 1985–1992, 1995
Elsevier Science Ltd

0031-3203/95 $9.50 + .00

0031-3203(95)00057-7

A FAST DIGITAL RADON TRANSFORM—AN EFFICIENT MEANS FOR EVALUATING THE HOUGH TRANSFORM

W. A. GÖTZ* and H. J. DRUCKMÜLLER
Institute for Computer Science, University of Innsbruck, Austria

(*Received* 27 *July* 1994; *received for publication* 7 *April* 1995)

Abstract—A fast digital Radon transform based on recursively defined digital straight lines is described, which has the sequential complexity of $N^2 \log N$ additions for an $N \times N$ image. This transform can be used to evaluate the Hough transform to detect straight lines in a digital image. Whilst a parallel implementation of the Hough transform algorithm is difficult because of global memory access requirements, the fast digital Radon transform is vectorizable and therefore well suited for parallel computation. The structure of the fast algorithm is shown to be quite similar to the FFT algorithm for decimation in frequency. It is demonstrated that even for sequential computation the fast Radon transform is an attractive alternative to the classical Hough transform algorithm.

Image processing Line detection Hough transform Radon transform
Fast algorithms

1. INTRODUCTION

Straight line detection is a basic task in many image-processing applications. The problem is usually solved by the Hough[1] transform (HT), a method that exploits the duality between points of a line and parameters of that line. A point in the image space is represented by a curve in the parameter space and lines of collinear points intersect in the parameter space at one point. These intersections are counted in an array of accumulators that quantizises the parameter space appropriately. So a probable line can be detected by searching for local maxima in the parameter space. The advantage of the HT is its stability against noise arising in digitized images. Since the HT is the Radon transform (RT) of a binary function, line detection can also be performed by a discrete RT. With use of the Fourier-slice theorem [e.g. Deans[2], Hall *et al.*[3]] and the FFT it is possible to compute a discrete RT in $O(N^2 \log N)$ multiplications for a $N \times N$ image. However, methods based on this theorem require interpolation from a rectangular grid into a circular grid which, for good quality, increases the computation time considerably. We introduce a fast algorithm for a digital RT based on a recursive representation of digital straight lines. Due to recursion the computational effort is $O(N^2 \log N)$ additions. It is shown, that the algorithm is vectorizable and therefore well suited for parallel computation. After deriving the fast digital RT we present a theoretical comparison between our fast algorithm and the classical HT-algorithm based on the same quantization of the parameter space. While the computational effort of the classical HT-algorithm mainly depends on the number of edge pixels, the fast digital RT is independent of this and consequently, a noisy image does not increase the computational time. Finally we demonstrate by a practical example the applicability of our transform for straight line detection.

2. RADON TRANSFORM

Before we introduce a discrete RT, it is useful to give a brief introduction to the analytical RT. We define the RT for the 2-D case, however, more general descriptions can be found e.g. in Deans.[2] Usually the RT is defined on the unit circle, but for our work it is more convenient to define it on the unit square.

Let $s, t \in \mathbf{R}$, $\theta \in [0, \pi]$ and f be a function with the unit square as its support, so that

$$f: \mathbf{R}^2 \to \mathbf{R} \quad \text{where } (x, y) \notin [0,1]^2 \Rightarrow f(x, y) = 0. \quad (1)$$

Using the unit vectors characterized by θ

$$\mathbf{e}(\theta) = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}, \quad \mathbf{e}^{\perp}(\theta) = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix} \quad (2)$$

the RT is defined as

$$\check{f}(\theta, t) = \mathscr{R}f = \int_{\mathbf{R}} f(t\mathbf{e}(\theta) + s\mathbf{e}^{\perp}(\theta))\, ds$$
$$= \int_{\mathbf{R}} f(t\cos(\theta) - s\sin(\theta), t\sin(\theta) + s\cos(\theta))\, ds. \quad (3)$$

Geometrically interpreted $\check{f}(\theta, t)$ is an integral along a line with normal distance t and angle θ. A line parametrized in that way is called the Hesse normal

* To whom correspondence should be addressed at Institut für Informatik, Universität Innsbruck, Technikerstraße 25/7, A-6020 Innsbruck, Austria

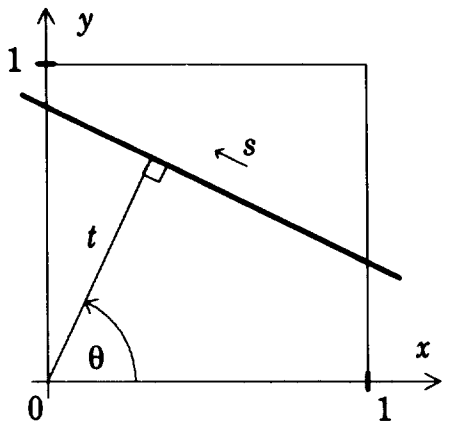

Fig. 1. Geometrical interpretation of the RT.

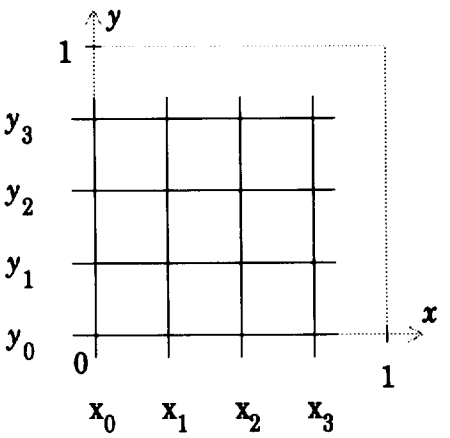

Fig. 2. A 4 × 4 grid.

form, where

$$t = x\cos(\theta) + y\sin(\theta). \tag{4}$$

This situation is outlined in Fig. 1.

Let us now consider a line in its slope/offset form

$$y = \sigma x + c \tag{5}$$

where σ is the slope and c is the offset of a straight line. It is obvious that a vertical line can not be expressed in this representation. Defining the RT with (σ, c) representation we assume $\sigma \in [0, 1]$. Of course now the set of possible lines is incomplete but this restriction is not relevant for the time being. With this limitation a sample of the RT is given by

$$\check{f}(\sigma, c) = \sqrt{1+\sigma^2}\int_0^1 f(x, \sigma x + c)\,dx. \tag{6}$$

3. DISCRETIZATION

Let f be a Function as defined in (1) sampled on a rectangular $N \times N$ grid by the following grid-points

$$x_i = \frac{i}{N}, \quad y_j = \frac{j}{N} \quad \text{where } i, j \in \{0, 1, \ldots, N-1\}. \tag{7}$$

An instance of such a grid is depicted in Fig. 2. Further we consider $f(x_i, y_j)$ as a pixel and the set of pixels as an image f. For simplicity $f(i, j)$ is written instead of $f(x_i, y_j)$.

In order to perform a RT on a function sampled on that grid one has to consider two steps:

- The appropriate quantization of the line-parameters.
- The approximation of each line by the available grid-points.

Performing the first step we define a line by two points of the grid, the origin (0,0) and the point $(N-1, s)$ where the parameter s is coupled with the slope of a line by the following equation

$$\sigma = \frac{s}{N-1} \quad \text{where } s \in \{0, 1, \ldots, N-1\}. \tag{8}$$

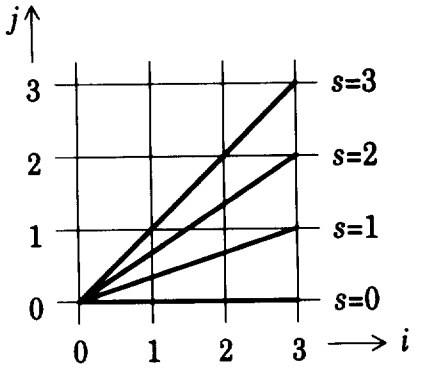

Fig. 3. Lines with parameter s in a 4 × 4 grid.

In Fig. 3 these lines are illustrated. Note that this is the finest possible resolution for the slope of a straight line in a $N \times N$ grid.

Until now we have only two points of a line. This is sufficient for an analytical line, but now we also have to provide the grid-points that approximate the analytical line. Such a line is called a digital straight line. The most obvious way to get a digital straight line is to round an analytical line to its nearest grid-points. It is clear that this would minimize the discretization error. In spite of this we present another definition of digital straight lines and show that with these lines a discrete RT can be performed very efficiently. For this reason the size of our image has to be a power of two, $N = 2^n$, $n \in \mathbf{N}$. For every natural number $0 \le i < 2^n$ we consider its binary representation $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \in \mathbf{Z}_2^n$ ($\mathbf{Z}_2$ is the dyadic group with addition and multiplication modulo 2) and the mapping $i = \lambda(u) = u_0 + 2u_1 + \cdots + 2^{n-1}u_{n-1}$. Now we can express every point in the grid (i, j) as $(\mathbf{u}, \mathbf{v}) \in \mathbf{Z}_2^n \times \mathbf{Z}_2^n$ where $i = \lambda(\mathbf{u})$, $j = \lambda(\mathbf{v})$. Instead of rounding the analytical line to the nearest grid-points we define a digital line as a mapping $l_s^n : \mathbf{Z}_2^n \to \mathbf{Z}$, where a line is defined recursively by

$$l_s^0(\mathbf{u}) = 0.$$

$$l_s^n(\mathbf{u}) = l_{\lfloor s/2 \rfloor}^{n-1}(u_0, \ldots, u_{n-2}) + \left\lfloor \frac{s+1}{2} \right\rfloor u_{n-1}. \tag{9}$$

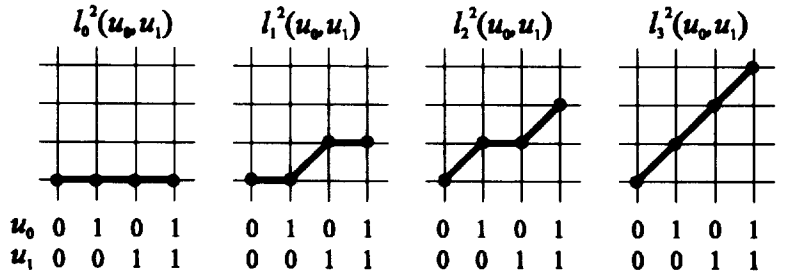

Fig. 4. Digital lines marked with black dots, $N = 4$.

(with $\lfloor\ \rfloor$ the integer part function denoting.) Roughly speaking a line with length N is composed of two lines with length $N/2$. Both of these lines have the same slope characterized by $\lfloor s/2 \rfloor$. Figure 4 shows the digital lines for a 4 × 4 grid.

An important question is how well our digital lines approximate the analytical lines. In Götz[18] a worst-case bound for the absolute distance between our digital lines and the corresponding analytical lines is derived. Let $j := l_s^n(\mathbf{u})$ and $i := \lambda(\mathbf{u})$ then we can state

$$|E_1| := \left| \frac{s}{N-1} x_i - y_i \right| \leq \frac{n+2}{4N}. \tag{10}$$

In other words the discretization error is of the order $O(\log N/N)$ referred to the unit square. Probably the bound (10) is not minimal because an empirical computed bound is $n/6N$ although we do not have a formal proof. Figure 5 illustrates a distribution for the error E_1 approximated by a histogram. It shows that small deviations are more likely than high ones which is a rather neat feature.

Lines, that do not pass through the origin of the grid can be expressed by a shifting parameter d. So a complete description of a digital line is $l_s^n(\mathbf{u}) + d$.

For an approximation of the analytical RT using our digital lines we first give the RT with parameters (s, d) which is easily derived from equation (6)

$$\check{f}(s,d) = \Delta(s) \int_0^1 f\left(x, \frac{s}{N-1} x + \frac{d}{N} \right) dx \tag{11}$$

where

$$\Delta(s) = \sqrt{1 + \left(\frac{s}{N-1} \right)^2}. \tag{12}$$

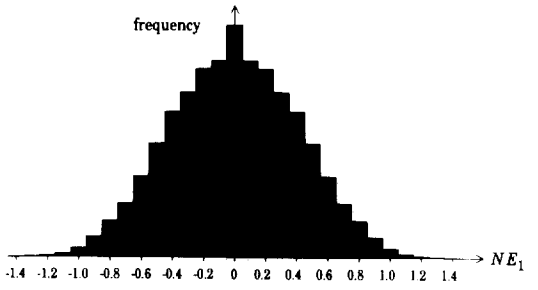

Fig. 5. Distribution of the error E_1 over all s, $N = 256$.

Now we define a digital Radon transform (DRT) as

$$\tilde{f}(s,d) = \sum_{\mathbf{u} \in Z_2^n} f(\lambda(\mathbf{u}), l_s^n(\mathbf{u}) + d), \tag{13}$$

and an approximation of the analytical RT as

$$\check{f}^{dis}(s,d) = \frac{\Delta(s)}{N} \tilde{f}(s,d). \tag{14}$$

Note that because of (1) and (7)

$$d \notin [-s, \ldots, N-1] \Rightarrow \tilde{f}(s,d) = 0. \tag{15}$$

In the rest of this paper we solely investigate the DRT because for line detection the factor $\Delta(s)$ is irrelevant. One algorithm to compute the DRT consists of determining the points of a line and then summing up all pixels placed on that line. Having N slopes and between N and $2N-1$ displacement per slope this would require $O(N^3)$ additions. In Table 1 and Table 2 an example for a 4 × 4 image is presented.

In the next section we derive a fast algorithm that performs the same task in $O(N^2 \log N)$ additions. At this point we want to remark that there are other types of fast discrete RTs given by Beylkin[4] and by Yang,[5] but these transforms require periodic images and are not useful for line detection purposes.

Table 1. $f(i,j)$

$j \backslash i$	0	1	2	3
3	1	0	0	1
2	0	0	1	0
1	1	1	0	1
0	1	0	0	0

$N = 4$.

Table 2. $\tilde{f}(s,d)$

$d \backslash s$	0	1	2	3
3	2	1	1	1
2	1	1	0	0
1	3	3	3	1
0	1	2	2	4
−1		0	1	0
−2			0	1
−3				0

4. FAST DIGITAL RADON TRANSFORM

The DRT in equation (13) can be separated into two components corresponding to $u_{n-1}=0$ and $u_{n-1}=1$, respectively.

$$\begin{aligned}\tilde{f}(s,d) &= \sum_{\mathbf{u}\in\mathbf{Z}_2^n} f(\lambda(\mathbf{u}), l_s^n(\mathbf{u})+d)\\ &= \sum_{\mathbf{u}\in\mathbf{Z}_2^{n-1}} f(\lambda(u_0,\ldots,u_{n-2},0), l_s^n(u_0,\ldots,u_{n-2},0)+d)\\ &\quad + \sum_{\mathbf{u}\in\mathbf{Z}_2^{n-1}} f(\lambda(u_0,\ldots,u_{n-2},1), l_s^n(u_0,\ldots,u_{n-2},1)+d)\end{aligned} \tag{16}$$

Using the recursive relation in (9) we obtain

$$\begin{aligned}\tilde{f}(s,d) &= \sum_{\mathbf{u}\in\mathbf{Z}_2^{n-1}} f(\lambda(u_0,\ldots,u_{n-2},0), l_{\lfloor s/2\rfloor}^{n-1}(\mathbf{u})+d)\\ &\quad + \sum_{\mathbf{u}\in\mathbf{Z}_2^{n-1}} f\left(\lambda(u_0,\ldots,u_{n-2},1), l_{\lfloor s/2\rfloor}^{n-1}(\mathbf{u}) + \left\lfloor\frac{s+1}{2}\right\rfloor + d\right)\end{aligned} \tag{17}$$

From this formula we can see that the first sum is the DRT of the left image half and the second sum is the DRT of the right including a shifting, thus the DRT of the entire image can easily be obtained by the DRTs of the two image halves. If this process is continued recursively it leads to a fast algorithm. Note that the decomposition (17) is quite analogous to the FFT algorithm of decimation in frequency [e.g. Nussbaumer[6]].

For a mathematical description of the fast algorithm for the DRT we use a formalism, which was introduced by Oberst[7] for the FFT. Let us define a mapping for $m\in\{0,1,\ldots,n\}$,

$$\begin{aligned}\tilde{f}^m(\mathbf{v},j) &= \tilde{f}^m((v_0,\ldots,v_{m-1},v_m,\ldots,v_{n-1}),j)\\ &= \sum_{\mathbf{u}\in\mathbf{Z}_2^m} f(\lambda(u_0,\ldots,u_{m-1},v_m,\ldots,v_{n-1}),\\ &\qquad l^m_{\lambda(v_m,\ldots,v_0)}(\mathbf{u})+j).\end{aligned} \tag{18}$$

This mapping describes the DRTs for 2^{n-m} strips of the image with size 2^m. Note that $\tilde{f}^0(\mathbf{v},j)=f(\lambda(\mathbf{v}),j)$ is the original and $\tilde{f}^n(\mathbf{v},j)=\tilde{f}(v_{n-1},\ldots,v_1,v_0),j)$ is the bit-reversal representation of its DRT. Using equation (15) it can be shown that $\tilde{f}^m(\mathbf{v},j)=0$ if $j\notin[-\lambda(v_{m-1},\ldots,v_1,v_0),N-1]$. Let us now derive a partial RT which maps $\tilde{f}^m(\mathbf{v},j)$ into $\tilde{f}^{m+1}(\mathbf{v},j)$. With this mapping the DRT can be computed by a composition of n partial RTs followed by a bit-reversal permutation step. We start with the definition of $\tilde{f}^{m+1}(\mathbf{v},j)$ and we will point out its relation to $\tilde{f}^m(\mathbf{v},j)$.

$$\begin{aligned}\tilde{f}^{m+1}(\mathbf{v},j) &= \sum_{\mathbf{u}\in\mathbf{Z}_2^m} f(\lambda(u_0,\ldots,u_m,v_{m+1},\ldots,v_{n-1}),\\ &\qquad l^{m+1}_{\lambda(v_m,\ldots,v_0)}(\mathbf{u})+j)\end{aligned} \tag{19}$$

This equation can again be separated into two components corresponding to $u_m=0$ and $u_m=1$, respectively.

$$\begin{aligned}\tilde{f}^{m+1}(\mathbf{v},j) &= \sum_{\mathbf{u}\in\mathbf{Z}_2^m} f(\lambda(u_0,\ldots,u_{m-1},0,v_{m+1},\ldots,v_{n-1}),\\ &\qquad l^{m+1}_{\lambda(v_m,\ldots,v_0)}(u_0,\ldots,u_{m-1},0)+j)\\ &\quad + \sum_{\mathbf{u}\in\mathbf{Z}_2^m} f(\lambda(u_0,\ldots,u_{m-1},1,v_{m+1},\ldots,v_{n-1}),\\ &\qquad l^{m+1}_{\lambda(v_m,\ldots,v_0)}(u_0,\ldots,u_{m-1},1)+j)\end{aligned} \tag{20}$$

Now we decimate the digital lines in equation (20) for one dimension. With $\lambda(v_m,\ldots,v_0)=v_m+2v_{m-1}+\cdots+2^m v_0$ we can observe, that

$$\left\lfloor\frac{\lambda(v_m,\ldots,v_0)}{2}\right\rfloor=\lambda(v_{m-1},\ldots,v_0), \tag{21}$$

and

$$\left\lfloor\frac{\lambda(v_m,\ldots,v_0)+1}{2}\right\rfloor=v_m+\lambda(v_{m-1},\ldots,v_0). \tag{22}$$

Using equation (21), (22) and the recursive relation in (9) we obtain

$$\begin{aligned}l^{m+1}_{\lambda(v_m,\ldots,v_0)}(u_0,\ldots,u_m) &= l^m_{\lambda(v_{m-1},\ldots,v_0)}(u_0,\ldots,u_{m-1})\\ &\quad +(v_m+\lambda(v_{m-1},\ldots,v_0))u_m.\end{aligned} \tag{23}$$

Using this result in equation (20) gives

$$\begin{aligned}\tilde{f}^{m+1}(\mathbf{v},j) &= \sum_{\mathbf{u}\in\mathbf{Z}_2^m} f(\lambda(u_0,\ldots,u_{m-1},0,v_{m+1},\ldots,v_{n-1}),\\ &\qquad l^m_{\lambda(v_{m-1},\ldots,v_0)}(\mathbf{u})+j)\\ &\quad + \sum_{\mathbf{u}\in\mathbf{Z}_2^m} f(\lambda(u_0,\ldots,u_{m-1},1,v_{m+1},\ldots,v_{n-1}),\\ &\qquad l^m_{\lambda(v_{m-1},\ldots,v_0)}(\mathbf{u})+j+v_m+\lambda(v_{m-1},\ldots,v_0))\end{aligned} \tag{24}$$

The first sum is the definition of $\tilde{f}^m(\mathbf{v},j)$ where the bit $v_m=0$. In the second sum $v_m=1$ and a shifting of $v_m+\lambda(v_{m-1},\ldots,v_0)$ is included. Let $\mathbf{v}^0:=(v_0,\ldots,v_{m-1},0,v_{m+1},\ldots,v_{n-1})$, $\mathbf{v}^1:=(v_0,\ldots,v_{m-1},1,v_{m+1},\ldots,v_{n-1})$ and $k:=\lambda(v_{m-1},\ldots,v_0)$, then the partial RT that maps $\tilde{f}^m(\mathbf{v},j)$ into $\tilde{f}^{m+1}(\mathbf{v},j)$ is given by

$$\tilde{f}^{m+1}(\mathbf{v},j)=\tilde{f}^m(\mathbf{v}^0,j)+\tilde{f}^m(\mathbf{v}^1,j+v_m+k). \tag{25}$$

It is obvious that the partial RT can be performed for all j in parallel. In order to derive a vectorized algorithm let us consider all relevant values $\tilde{f}^m(\mathbf{v},j)$ as one vector $\tilde{\mathbf{f}}^m(\mathbf{v})$ and define a $(2N-1)\times(2N-1)$ shifting-matrix as follows

$$\tilde{\mathbf{f}}^m(\mathbf{v})=\begin{pmatrix}\tilde{f}^m(N-1)\\ \vdots\\ \tilde{f}^m(-N+1)\end{pmatrix},\quad \mathbf{S}=\begin{pmatrix}0 & & \cdots & & 0\\ 1 & 0 & & & \\ 0 & 1 & & & \vdots\\ \vdots & & 1 & 0 & \\ 0 & \cdots & 0 & 1 & 0\end{pmatrix}. \tag{26}$$

(Note that multiplication with $\mathbf{S}^k$ causes a shifting of k lines downwards.) Thus we can express (25) in matrix

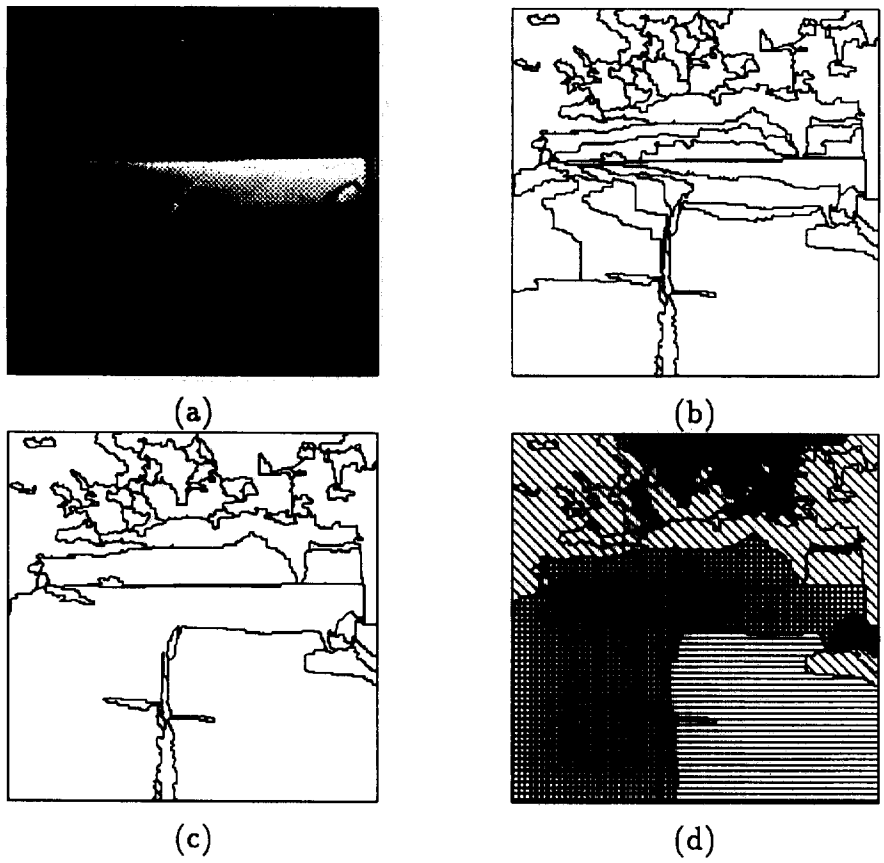

Fig. 1. An example of mislabeling results due to mis-segmentation: (a) a scene, (b) boundaries of an initial set of segmented regions, (c) boundaries of an enhanced set of segmented regions and (d) labeling results (see Fig. 3 for the meaning of the pattern keys).

regions are misinterpreted [Fig. 1(d)] when we applied our MRF-based image interpretation scheme.[14] Therefore, if once we merge these two regions into one based on some evidence or strategies, the regions corresponding to the *road* and *roadline* might be correctly labeled.

So far, most of the region labeling systems have tried to overcome such erroneous segmented regions by using the initial labels of and compatibilities of labels between these regions.[2,3,5] That is, they usually divided the given image into a set of small fragments and continuously merged them based on their spatial adjacencies and equality of assigned labels. However, in that paradigm, the labeling process wholly depends on the unary properties of and spatial adjacencies between regions and may give much erroneous interpretation results because the unary properties determining the initial labels are quite unreliable in natural scenes and the initial set of regions segmented into small fragments has many ambiguities in relative spatial relationships between regions.

2.2. *The global scheme*

The proposed integrated scheme is shown in Fig. 2. In Fig. 2 the left-half and the right-half of the figure correspond to the image segmentation and interpretation processes, as already described in our previous works,[13,14] respectively.

As before, we start with the initial set of segmented regions by applying the ISM (Initial Segmentation Module), which is implemented as one of the traditional region-based image segmentation techniques. In the integrated scheme, the interpretation process is then applied to the temporal set of region clusters which are formed from the current set of region labels assigned to the initial set of segmented regions by the segmentation process. The optimality of the current sets of region labels for segmentation and interpretation is then jointly estimated through a unified energy function.

However, at the initial stage of the optimization process, the current set of region clusters may be far from the ideal segmentation state, which indispensably misleads the interpretation process. Also, the misinterpretation labels for the current set of region clusters prevent the segmentation process from proceeding to the more optimal segmentation state via the unified energy function. Thus, we should control the weights between two processes as the optimization process proceeds. That is, at the initial stage of the optimization process, we given more attention to the segmen-

A detailed analysis of the HT discretization errors was given by van Veen and Groen.[14]

The computational effort of the HT-algorithm depends at the one hand on the resolution of the grid and on the other hand on the number of edge pixels. Let p be the number of edge pixels, then the number of accumulator increments is given by

$$a_2 = pN. \tag{32}$$

Usually, for an edge-detected binary image p is much less than N^2.

6. FDRT VERSUS HT-ALGORITHM

We compare the two algorithms simply in terms of the number of relevant additions. This theoretical measure is chosen because we want to neglect control commands depending on the implementation (for example, the multiplication σx can be replaced by a table-look up). From that point of view we can state that the FDRT is faster than the HT-algorithm if $a_2 < a_1$. So the quotient

$$\frac{a_1}{a_2} \approx \frac{p}{N(n+1)} \tag{33}$$

gives the theoretical speed up of the FDRT versus the HT-algorithm. In other words, when a binary image has more than $N(n+1)$ edge pixels, the FDRT is faster. Suppose $N = 256$ then this bound corresponds to nine image lines. In practice this condition is often fulfilled, because usually a line detection process consists of the following steps:

(1) Generate a grey level image using a video camera or a scanner.

(2) Amplify the edges in the image, e.g. using the Sobel-operator [e.g. Gonzalez and Woods[8]].

(3) Generate a binary image using a certain threshold.

(4) Perform the HT-algorithm.

(5) Evaluate the parameter space.

A digitized line produced in this way has more than one pixel width and usually also some noise arises in the binary image. This fact causes two important arguments for the FDRT.

- Since the HT-algorithm depends on the number of edge pixels, a noisy image increases the computation time. But this is not relevant for the FDRT, because its speed is independent of the number of edge pixels. Of course a thinning and noise reduction process can reduce the number of edge pixels but this step also needs some effort.
- A digital line as used in the FDRT counts pixels more distant from the ideal (analytical) line than the HT, but if the width of a line is several pixels, many of these counts still fall within the scope of the digitized line. Thus the higher discretization error implicit in the FDRT has little influence. Certainly, this observation should be analysed more accurately in further work.

Indeed, the most substantial advantage of the FDRT lies in our opinion in the potential for parallel computation. The HT-algorithm is highly parallel, but therefore the array of accumulators must be a common memory for the parallel processing elements. This is a strong restriction for currently available hardware. Several algorithms using different computer architectures have been proposed for parallel implementation of the HT [e.g. Sanz *et al.*,[15] Blanford[16] and Cypher and Sanz[17]]. As mentioned before the FDRT is vectorizable and little common memory access is necessary. This feature makes it easy to implement the FDRT on various parallel-computers. Due to its similarity to the FFT, it should be easy to run the FDRT on computer architectures that support the butterfly structure. However, until now we have not investigated that topic seriously. Notice that it is also possible to utilize parallelism in a conventional computer for example with 32-bit architecture. If one wants to compute the FDRT for f and f_1, then the first 16 bits can be used for computing $\tilde{f}$ and the remaining 16 bits for computing $\tilde{f}_1$, respectively.

7. EXAMPLE

The (sequential) algorithms for the following example were implemented in the C language on a HP 710 workstation. For the HT-algorithm the multiplications were replaced by a look-up table. Figure 8(a) shows a 256×256 grey level image where three straight lines, detected by our method are overlaid. Bright areas correspond to a high grey level and dark areas to low grey level. Figure 8(b) shows a binary image as a result of applying the Sobel-operator followed by a threshold operation. The Radon space of the binary image computed by the FDRT is presented in Fig. 8(c). There is no visual difference when it is computed using the HT-algorithm. The quotient

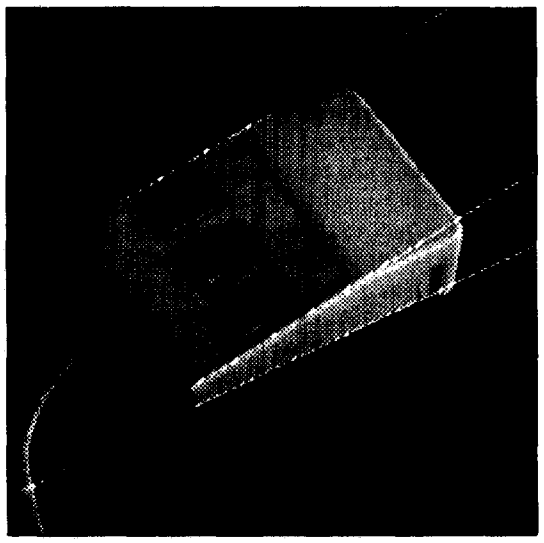

Fig. 8(a). A 256×256 grey level image.

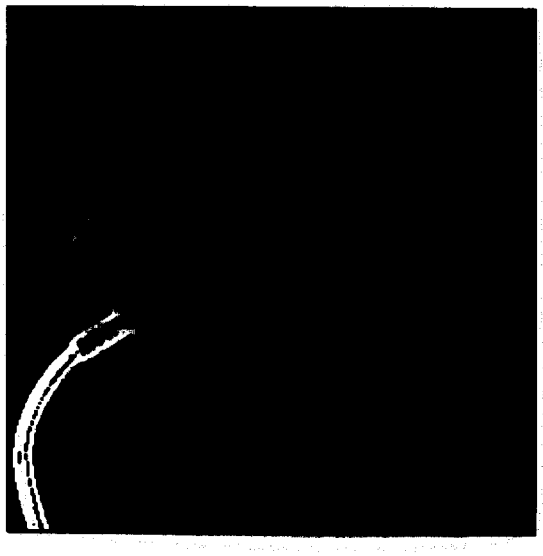

Fig. 8(b). Binary image produced by the Sobel operator followed by a threshold operation.

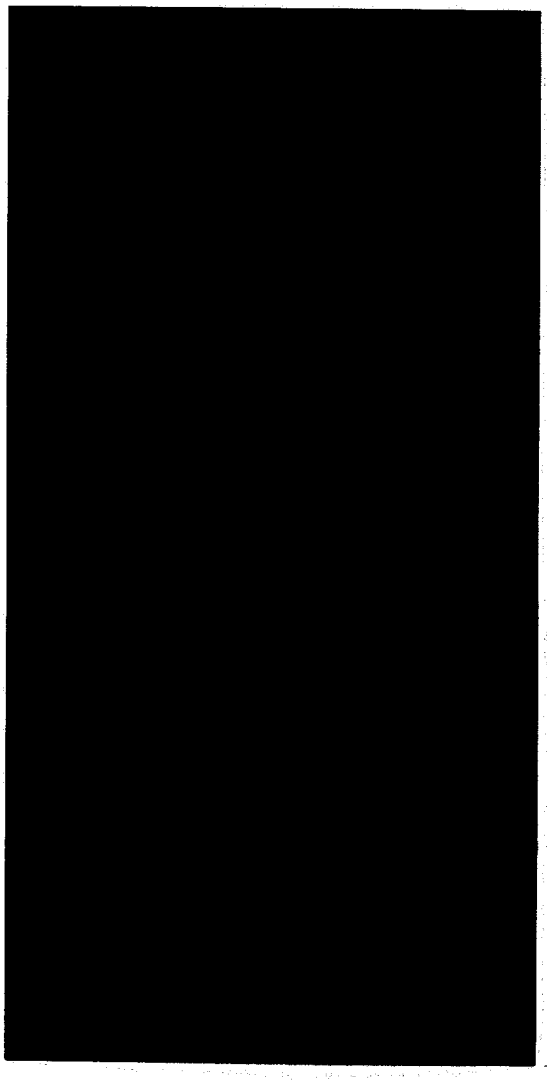

Fig. 8(c). Radon space produced by the FDRT.

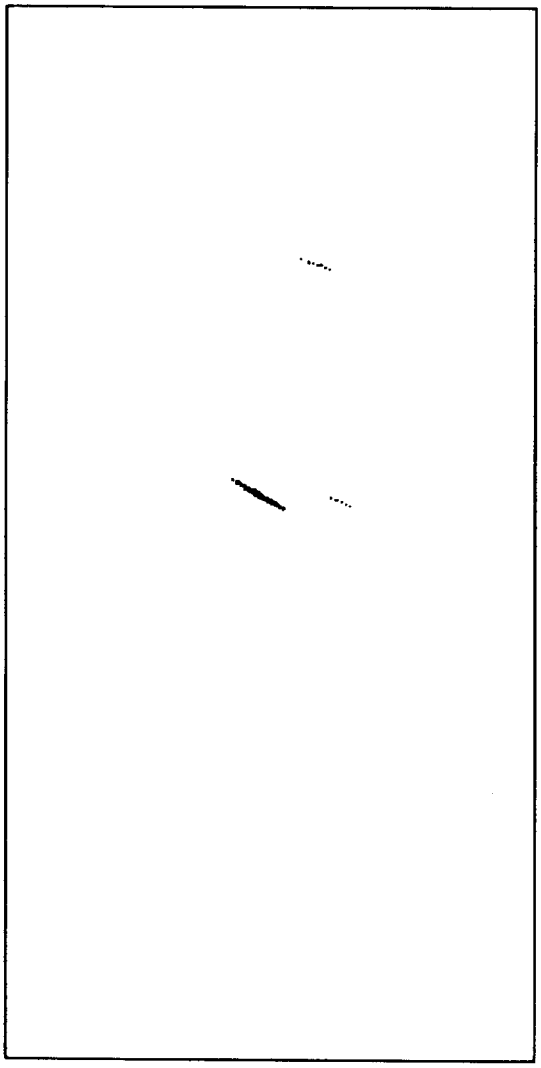

Fig. 8(d). Clusters detected by a threshold of $N/3$.

$p/N(n+1) = 1.88$ means that the HT-algorithm has about two times more relevant additions than the FDRT. This theoretical result is actually improved upon in the measured computation time which shows that the FDRT is 2.78 times faster than the HT-algorithm. For the evaluation of the Radon space a threshold of $N/3$ was chosen to omit shorter lines to be detected [Fig. 8(d)]. The resulting clusters were analysed by a simple minimum-distance classifier[8] and the centre of each cluster was used to determine the corresponding line. These lines are overlaid on the original image in Fig. 8(a).

Although our method has a very high resolution this carries one shortcoming with it. Van Veen and Groen[14] pointed out, that with increasing resolution of the parameter space also the extension of line peaks is increasing [see Fig. 8(d)]. This makes the detection of local maxima more difficult, but still tractable. It will be a future task to develop an optimal method to evaluate the Radon space for our fixed geometry.

Clearly, these results strongly depend on our image and on the implementation of the algorithms, but the principal usefulness of our methods was demonstrated. In other cases the classical HT-algorithm is still preferable, especially when a lower or limited quantization of the parameter space is sufficient or the number of edge pixels is very small.

8. CONCLUSION

A fast digital Radon transform was presented based on a recursive definition of digital straight lines. The transform algorithm parallels the development of the FFT algorithm for decimation in frequency and can be expressed by a butterfly-structure. The algorithm is vectorizable and therefore well qualified for parallel computation. A new balance between computation complexity and discretization error was achieved for the same quantization of the parameter space. The computation complexity was reduced from $O(N^3)$ to $O(N^2 \log N)$ and the discretization error was increased from $O(1/N)$ to $O(\log N/N)$ for a $N \times N$ image. It was demonstrated, that for line detection purposes the slightly higher discretization error has little influence, but the saving in computation time can be considerable.

Acknowledgements—The authors wish to thank Prof. U. Oberst and Prof. N. C. Steele for their helpful comments and stimulating discussions. Sections 1–4 of this paper are an abridged version of the first author's doctoral dissertation which was written under supervision of Prof. U. Oberst, Innsbruck.

REFERENCES

1. V. C. Hough, Method and means for recognizing complex patterns, U.S. Patent 3069654 (1962).
2. R. D. Deans, *The Radon Transform And Some Of its Applications*. John Wiley & Sons (1983).
3. Hall T. J. Terell, J. M. Senior and L. M. Murphy, A new fast discrete Radon transform for enhancing linear features in noisy images, *IEE Conf. Public.*, Third International Conference on Image Processing and its Applications, 187–191 (1989).
4. G. Beylkin, Discrete Radon transform, *IEEE Trans. Acoust. Speech Sig. Proc.* **ASSP-35**, 162–172 (1987).
5. D. Yang, Fast discrete Radon transform and 2-D discrete fourier transform, *Electr. Lett.* **26**, 550–551 (1990).
6. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*. Springer Verlag, (1981).
7. U. Oberst, The fast Fourier transform, lecture notes, University of Innsbruck, (1991).
8. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, (1992).
9. D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* **13**, 111–122 (1981).
10. S. Davis, Hierarchical generalized Hough transforms and line-segment based generalized Hough transforms, *Pattern Recognition* **15**, 277–285 (1982).
11. M. A. Li Lavin and R. J. Le Master, Fast Hough transform: a hierarchical approach, *Comp. Vision Graph. Image Process.* **36**, 139–161 (1986).
12. J. Illingworth and J. Kittler, The adaptive Hough transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**, 691–698 (1987).
13. D. Svalbe, Natural representation for straight lines and the Hough transform on discrete arrays, *IEEE Trans. Pattern Anal. Mach. Intell.* **11**, 941–950 (1989).
14. M. van Veen F. C. A. Groen, Discretization errors in the Hough transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 137–145 (1981).
15. L. C. Sanz and E. B. Hinkle and A. K. Jain, *Radon and Projection Transform-Based Computer Vision. Springer Verlag, (1988).*
16. P. Blanford, Dynamically Quantized Pyramids for Hough Vote Collection, *Technical Report* #87-03-08, University of Washington (March 1987).
17. E. Cypher, J. L. C. Sanz, The Hough transform has $O(N)$ complexity on SIMD $N \times N$ Mesh Array Architectures, *Proc. of IEEE* CAPAMI Workshop (1987).
18. W. A. Götz, *Eine Schnelle Diskrete Radon Transformation basierend auf rekursiv definiertern Digitalen Geraden*, Dissertation, University of Innsbruck, (1993).

About the Author—WALTER A. GÖTZ was born in 1965 in Kufstein, Austria. He received the Diploma-Ingenieur degree in computer science from University of Linz, Austria, in 1990 and the doctoral degree from University of Innsbruck, Austria, in 1993. From 1991 he is with the Institute for Computer Science, University of Innsbruck. His research interests are image processing and object oriented programming.

About the Author—HEINZ J. DRUCKMÜLLER was born in 1959 in Schwaz, Austria. He received the diploma degree in mathematics in 1981 and the doctoral degree in 1983, both from University of Innsbruck, Austria. From 1982 he is with the Institute for Computer Science, University of Innsbruck. His research interest is in the area of numerical analysis.