Constant-Working-Space Image Scan with a Given Angle

Tetsuo Asano [*]

Abstract

This paper proves that there is a linear-time algorithm for scanning an image with a given angle using only constant size of working space. This is an extension of a popular raster scan using $O(1)$ working space.

1 Introduction

There are increasing demands for highly intelligent peripherals such as printers, scanners, and digital cameras. To achieve intelligence they need sophisticated built-in or embedded softwares. One big difference from ordinary software in computers is little allowance of working space which can be used by the software. In this paper we propose a space-efficient algorithm used for image processing. Especially, we present an algorithm for scanning an image with any angle. It is rather easy to do it using only constant-size working space with some sacrifice of running time. This paper achieves the goal without any sacrifice of running time. That is, we present an algorithm for scanning an image consisting of n^2 pixels with an arbitrarily given angle using only constant-size working space which runs in $O(n^2)$ time. More important for embedded software is simpleness of an algorithm. In fact, it requires no sophisticated data structure or recursive function.

The work in this paper would open a great number of possibilities in applications to computer vision, computer graphics, and build-in software design. Image scan is just one of the most important topics in the direction.

2 Rotated Raster Scan

Let G be an image consisting of $n \times n$ pixels where each pixel has integer coordinates ranging from 0 to $n-1$. It is easy to enumerate all those pixels in a raster order, that is, in the order of $(0,0),(1,0),\ldots,(n-1,0),(0,1),(1,1),\ldots,(n-1,n-1)$. That is,

$$G = \{(x,y) \mid x,y = 0,1,\ldots,n-1\}. \quad (1)$$

This enumeration can be implemented by a program of double loops:

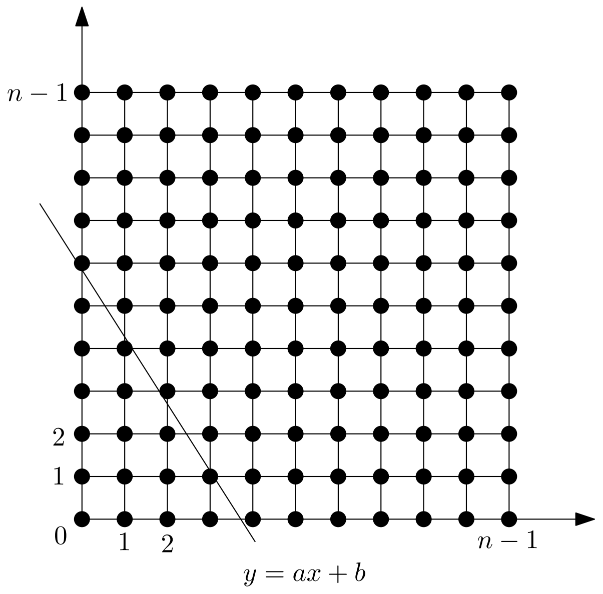

Figure 1: A square image and a line determining scan order.

```
for y = 0 to n-1 do
    for x = 0 to n-1 do
        output a pixel (x, y);
```

This scan is referred to as a row-major raster scan. In a similar way we can define a column-major raster scan. As an extension or generalization we consider a rotated raster scan in which pixels are enumerated along lines of a given angle. Throughout the paper we assume that the angle of those lines is given as a slope a instead of angle and we call the line $y = ax$ the **guide line** for the rotated scan. Figure 1 shows an example of an image and a guide line specifying scan order.

Throughout the paper we assume that the slope a of the guide line is negative to simplify the argument. The case of a positive slope is symmetric. We start from the pixel $(0,0)$. The next point is either $(0,1)$ or $(1,0)$ depending on which is closer to the line $y = ax$ of slope a passing through the origin $(0,0)$. In this way we output pixels in the increasing order of the vertical distances to the line $y = ax$. Therefore, we can describe the rotated raster scan using a priority queue PQ.

Rotated Raster Scan with Slope a
— Algorithm 1:

[*]School of Information Science, JAIST, Japan, t-asano@jaist.ac.jp

```
PQ: priority queue keeping pixels with vertical
distances to the line $y = ax$ as keys.
for $x = 0$ to $n - 1$
    Put a pixel $(x, 0)$ into PQ with key = $-ax$.
repeat{
    Extract a pixel $(x, y)$ of the smallest key from PQ.
    Output the pixel $(x, y)$.
    if $(x, y) = (n - 1, n - 1)$ then exit from the loop.
    if $(y < n - 1)$ then put a pixel $(x, y + 1)$ into PQ
        with key = $y + 1 - ax$.
}
```

It is easy to see that the algorithm is correct and runs in $O(n^2 \log n)$ time using $O(n)$ working space. Correctness of the algorithm is based on the observation that in each column (of the same x coordinate) pixels are enumerated from bottom to top one at a time. Thus, whenever we output a pixel (x, y), we remove the pixel from the priority queue and insert the pixel just above it, i.e., $(x, y+1)$ if it is still in the image area G.

3 Scanning by Rational Slope

Recall that the objective of this paper is to reduce the space requirement by an algorithm for scanning an image with a given angle. As a first step we show it is possible to reduce the working space if a slope a of a guide line is a rational number $-q/p$ such that its numerator and denominator are both less than n, the image size, and the two integers p and q are prime to each other.

Now, we want to sort all the pixels in the increasing order of their vertical distances to the guide line $y = ax = -(q/p)x$. If we denote the vertical distance from a pixel (x, y) to the guide line by $d(x, y)$, then we have

$$d(x, y) = y + \frac{q}{p}x = (py + qx)/p. \tag{2}$$

Since p is fixed, sorting pixels by the values of $d(x, y)$ is equivalent to sorting them by the values $py + qx$.

Now, we partition a set of pixels by lines passing through the bottom pixels $(0, 0), (1, 0), \ldots$. That is, by P_k we denote a set of pixels which lie above the line $y = (-q/p)(x - k)$ and below $y = (-q/p)(x - k - 1)$, or more formally we define

$$\begin{aligned} P_k &= \{(x, y) \in G \mid \\ &\qquad (-q/p)(x-k) \le y < (-q/p)(x-k-1)\}, \\ k &= 0, 1, \ldots, r,\ r = \lfloor (n-1)(1+p/q) \rfloor. \end{aligned}$$

An easy observation here is that if we can order all pixels in each such set then it suffices to output pixels in the order of $P_0, P_1, \ldots, P_r$. Then, how can we order those pixels in a set P_k? We further partition the set of pixels in P_k by their y coordinates, that is, we define a set

$$\begin{aligned} P_k[m] &= \{(x, y) \in P_k \mid m \le y/q < m+1\}, \\ &\qquad m = 0, 1, \ldots, \lceil n/q \rceil \}. \end{aligned}$$

Figure 2 shows $P_6[0]$ and $P_6[1]$ in an image of size 11×11, where the guide line is $y = (-5/2)x$, i.e., $p = 2, q = 5$, and $n = 11$. The set $P_6[0]$ consists of $(6,0), (6,1), (6,2), (5,3), (5,4)$ and $P_6[1]$ consists of $(4,5), (4,6), (4,7), (3,8), (3,9)$. Because of the definition, if (x, y) is an element of $P_k[m]$ then $(x-p, y+q)$ is an element of $P_k[m+1]$ if $(x-p, y+q) \in G$. In the example of Figure 2, $(6,0) \in P_6[0]$ implies $(6-2, 0+5) = (4,5) \in P_6[1]$. More generally, if (x, y) is an element of the set $P_k[0]$ then the pixel $(x - jp, y + jq)$ belongs to the set $P_k[j]$ for any integer j if the pixel is within the image area, that is, if $0 \le x - jp < n$ and $0 \le y + jq < n$. Now we define a natural equivalence relation:

$$\begin{aligned} &(x, y) \equiv (x', y') \iff (x, y), (x', y') \in G \text{ and} \\ &x \equiv x' \mod p \text{ and } y \equiv y' \mod q. \end{aligned}$$

Two equivalent pixels have the same vertical distance to the guide line. In fact, for two equivalent pixels (x, y) and $(x - jp, y + jq)$ we have

$$\begin{aligned} d(x, y) &= (py + qx)/p = (p(y + jq) + q(x - jp))/p \\ &= d(x - jp, y + jq). \end{aligned}$$

This implies that once we can order elements of the set $P_k[0]$ then it is easy to order all the elements of the set P_k. Whenever we output a pixel (x, y) in $P_k[0]$, we should output its equivalent pixels in $P_k[1], P_k[2], \cdots$ in order before reporting the next pixel in $P_k[0]$.

Now consider how to order pixels in the set $P_k[0]$. Note that the x coordinates of those pixels are between k and $k - p + 1$. If there are two or more pixels in the same x coordinates then they are reported in the increasing y order. Thus, if a priority queue of size p is available, the previous mechanism can order the pixels in the set $P_k[0]$. Thus, we have the following lemma.

Lemma 1 *Given an $n \times n$ image G and a rational slope $a = -q/p$, there is an algorithm for enumerating all pixels in the order determined by the slope which runs in $O(n^2 \log p)$ time using $O(p)$ working space.*

Proof. The x-coordinates of the pixels in the set $P_k[0]$ range from k to $k - p + 1$ (the pixel $(k - p, q)$ belongs to $P_k[1]$). In each column j, the first pixel to be enumerated is the one just above the line $y = (-q/p)(x - k)$ passing through the pixel $(k, 0) \in P_k[0]$. Since the pixels in the columns $k-1, k-2, \cdots$ are $(k-1, \lceil q/p \rceil), (k-2, \lceil 2q/p \rceil), \cdots$. In general, the pixel at column $k - j$ is $(k - j, \lceil jq/p \rceil)$. Starting from those pixels, we repeatedly choose pixels in the increasing order of their vertical distance to the line $y = (-q/p)(x - k)$, that is, $\lceil jq/p \rceil + (q/p)(k - j - k) = \lceil jq/p \rceil - (jq/p)$ since p and q are prime to each other. Whenever we choose a pixel (x, y) then we remove it from the priority queue and then insert a pixel just above it, that is, $(x, y + 1)$ as far as it also belongs to the same set $P_k[0]$. Then, we report the pixel (x, y) and then all of its equivalent pixels in order.

The algorithm is as follows:

Rotated Raster Scan with Slope a — Algorithm 2:

```
PQ: priority queue keeping pixels with vertical
distances to the line $y = ax$ as keys.
for $k = 0$ to $n - 1$
    Enumerate all pixels in the set $P_k$ in order.
Enumeration of pixels in $P_k$.
    for $j = 0$ to $p - 1$
        put a pixel $(k - j, \lceil jq/p \rceil)$ into PQ with
        key $= \lceil jq/p \rceil - (jq/p)$.
    repeat{
        Extract a pixel $(x, y)$ having the smallest key out
        of PQ.
        do{
            Output the pixel $(x, y)$.
            $x = x - p, y = y + q$.
        } while($(x, y) \in G$)
        if $y < (-q/p)(x - k - 1)$
            then put a pixel $(x, y + 1)$ with
            key $= y + 1 + (q/p)(x - k)$.
    }
```

□

Of course, if $p = O(n)$ then this algorithm is not an improvement of the previous trivial algorithm which runs in $O(n^2 \log n)$ time using $O(n)$ working space. Now, a natural question is whether there is an algorithm which runs in linear time using only constant size of space. In the next section we present one such algorithm.

4 Linear-Time Constant-Working-Space Algorithm

Our goal here is to achieve constant size of working space while keeping the running time linear in the number of pixels. If the goal is just to achieve the constant size working space, then it is rather easy. Let (x_i, y_i) be the current pixel (its initial value is of course $(0, 0)$). To find the next pixel, for each column we compute the pixel just above the line $y = a(x - x_i) + y_i$ and measure the vertical distance from the pixel to the line. We just choose the pixel of the smallest vertical distance. If there is a tie, we prefer one of a smaller x coordinate. Since it is just

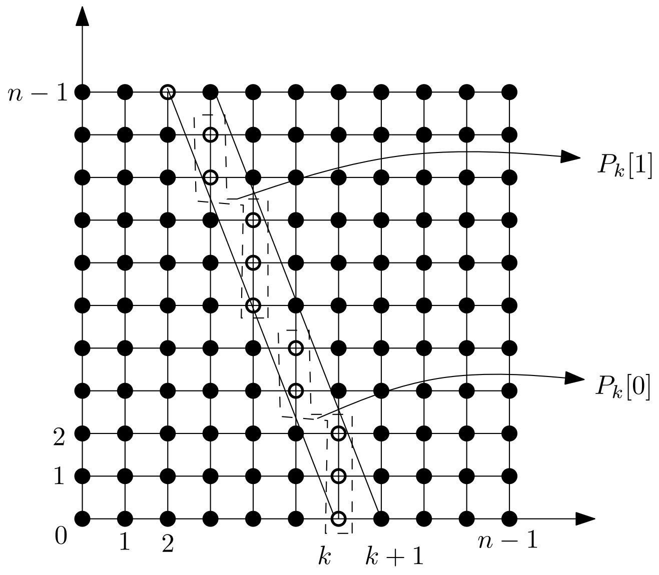

Figure 2: An image with a guide line $y = (-5/2)x$. Two sets of pixels, $P_6[0]$ and $P_6[1]$ are depicted by empty circles.

minimum-finding process, working space we need is just constant size. Of course, for each pixel we need $O(n)$ time to choose the next pixel, and thus $O(n^3)$ time in total.

Our question is whether we can implement the scan using constant-size working space without sacrificing the running time. Fortunately, the answer is yes. A key idea is to use integer properties. The algorithm itself is quite similar to Algorithm 2 described before. An important difference is that no priority queue is used to reduce the working space.

A key is how to order pixels in the set $P_k[0]$. It is known that the first pixel is $(k, 0)$. How can we know the next pixel without using a priority queue? As we have observed before, scan order coincides with the increasing order of the key values $py + qx$ in the set $P_k[0]$. The key value for the starting point $(k, 0)$ is qk. We want to know where is a pixel of key value $qk + i$ for $i = 1, 2, \ldots, p - 1$ (note that $P_k[0]$ consists of p pixels).

Suppose we are at a pixel (x, y). If we move to the left, that is, from (x, y) to $(x - 1, y)$, then the key decreases by q, and when we move to $(x, y + 1)$ then it increases by p. As far as we stay in the same column (x-coordinate), the value of key modulo p remains unchanged. To move to a pixel whose key value is different from the current key value by $i, 0 < i < p$, we have to move horizontally by d_i or $-(p - d_i)$, that is, to $(x + d_i, y)$ or $(x - (p - d_i), y)$, determined by

$$d_i q \equiv i \mod p. \tag{3}$$

Since either $(x + d_i, y)$ or $(x - (p - d_i), y)$ lies in the set $P_k[0]$, we can easily choose the right one. In practice,

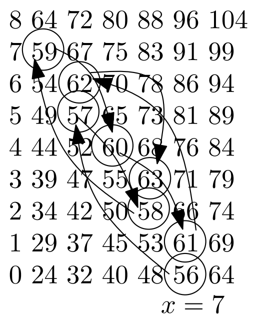

Figure 3: Ordering pixels in the set $P_7[0]$.

we do not need all of $d_1, \ldots, d_{p-1}$ since we want to find pixels of key values $kq, kq+1, \ldots, kq+p-1$ in an incremental fashion. So, it suffices to know the integer d_1 defined by

$$d_1 q \equiv 1 \mod p. \tag{4}$$

Once we know the x-coordinate and key value of the next pixel, it is easy to calculate its y-coordinate since we know that the key value is given by $py + qx$.

For example, when $p = 5, q = 8$, we have $d_1 = 2$ since $2q \mod p = 16 \mod 5 = 1$. When $k = 7$, the pixel $(7, 0)$ has a key 56. The next pixel is either at $x = 7 + 2 = 9$ or $x = 7 - (5 - 2) = 4$. Since $x = 9$ is outside $P_7[0]$, we can conclude that the next pixel must be at $x = 4$. Now we know the key value is 57. Thus, the y-coordinate is $(57 - 8 \times 4)/5 = 5$. In this way, we can find the next pixel without using a priority queue. See Figure 3.

The algorithm is as follows:

Rotated Raster Scan with Slope a
— Algorithm 3:

```
Calculate a smallest positive integer $d$ satisfying
    $dq \equiv 1 \mod p$.
for $k = 0$ to $n - 1$
    Enumerate all pixels in the set $P_k$ in order.
Enumeration of pixels in $P_k${
    report$(k, 0)$.
    $(x, y) = (k, 0)$.
    for $i = 1$ to $p - 1$ {
        if $x + d \leq k$ then $x' = x + d$
        else $x' = x - (p - d)$.
        $y' = (kq + i)/p$.
        report$(x', y')$.
        $(x, y) = (x', y')$.
    }
}
procedure report$(x, y)${
    do{
        Output the pixel $(x, y)$.
        $x = x - p, y = y + q$.
    } while($(x, y) \in G$)
}
```

Algorithm 3 needs a small modification to handle the part after the bottom right corner pixel $(n-1, 0)$, but it is omitted due to page limit.

Theorem 2 *Given an $n \times n$ image G and a rational slope $a = -q/p$, Algorithm 3 above correctly enumerates all the pixels in the order determined by the slope and runs in $O(n^2)$ time using $O(1)$ working space.*

5 Irrational Angles

So far we have considered the case where a slope is given as a rational number with its numerator and denominator less than image size. What happens if the slope is irrational? Fortunately, it is not so hard to adapt the proposed algorithm for the case. An important observation is that our guide line is well characterized by which two pixels should be separated. Thus, if we consider a finer grid defined by half grid gap, then such a line is characterized by a line passing through two grid points in the finer grid. Thus, it is not so hard to adapt our algorithm for the case.

6 Applications

Such a rotated scan is required for scanner technology. When we scan a document by a scanner, the document is sometimes rotated. Correction of such rotation is usually done in a scanner. Since scanners cannot possess much working space, space-efficient algorithms are desired. Asano et al. [1] propose a space-efficient algorithm for correcting such a rotation. Especially, it is an in-place algorithm which uses no extra working space other than a given image. In the algorithm a row-major or column-major raster scan is used to scan a rotated subimage. However, by our experience a rotated raster scan sensitive to rotation angle is more desirable.

Acknowledgments

This work of T.A. was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas and Scientific Research (B). The author would like to express his sincere thanks to Erik Demaine, Stefan Langerman, Ryuhei Uehara, Mitsuo Motoki, and Masashi Kiyomi for their stimulating discussions.

References

[1] T. Asano, S. Bitou, M. Motoki and N. Usui: "In-Place Algorithm for Image Rotation, "Proc. ISAAC 2007, pp. 704-715, Sendai, Dec. 2007.