AN EFFICIENT ALGORITHM FOR LINE RECOGNITION BASED ON INTEGER ARITHMETIC

KHALID DAGHAMEEN[*]
NABIL ARMAN[**]

[*]Department of Electrical and Computer Engineering
College of Engineering and Technology
Palestine Polytechnic University
Hebron, Palestine
dkhalid@ppu.edu

[**]Department of Mathematics and Computer Science
College of Applied Sciences
Palestine Polytechnic University
Hebron, Palestine
narman@ppu.edu

ABSTRACT

Line recognition is an important aspect in image processing. Many line detection algorithms were introduced. However these algorithms involve expansive operations like floating point arithmetic and matrix operations, which increase their computational time significantly. In this paper, a line recognition algorithm is introduced which involves integer arithmetic only.
Keywords: *Line recognition, line detection, image processing*

1. INTRODUCTION

Image processing is one of the most important areas in computer science and engineering. One of the main goals of image processing is to be able to identify objects of the image. Objects in a computer image are identified by their edges. These edges could be straight lines or curved lines. Many algorithms were used to detect a line in gray-scaled images as well as colored images. Each algorithm has some conditions and constraints to deal with. Guido introduces an algorithm to detect a line based on a weighted minimum mean square error formulations [1]. This algorithm works with matrices and uses a set of matrix operations such as transpose and multiplications.

Hough's transform is another algorithm for detecting a line. This algorithm is based on quantization and simple line equations, and finding the maxima. Hough algorithm is an efficient algorithm when the slope is small. On the other hand, it becomes inefficient or even impractical to use when the slope approaches infinity [2,6]. This algorithm is able to detect straight and curved lines, circles and ellipses.
Marco and others developed a set of algorithms to detect a line based on a general formulation of a combinatorial optimization problem. In their algorithm, a lot of expensive operations like multiplication, power and exponent functions are used. Marco and his colleagues claimed that they are able to compare their algorithms with Hough's transform algorithms and they claim that their algorithms are faster [3].

Guru developed an algorithm based on small eigenvalue analysis to detect straight line segments in an edge image. The algorithm depends on scanning the input image from top left corner to the bottom down corner with a moving mask matrix. The small eigenvalue of covariance of the edge pixels is computed. This algorithm depends on matrix processing [4].

A straight line detection algorithm was presented by Yun-Soak Lee and others [5]. It separates rows and column edges from edge image using primitive shapes, the edges are labeled and analysis for each edge is performed.

Chan and his colleagues used a simple and efficient approach for the line segment detection algorithm, which employs the properties of digital line segment. In discrete domain, a continuous line can be considered as combination of sets of patterns in the quantized direction of edge pixels (i.e. 0°, 45°, 90°, 135°) with

approximately equal number of pixels. Based on this characteristic, the boundary convergence of discrete line is derived. According to this convergence property, a simple and efficient line detection algorithm is proposed. Due to its simplicity and efficiency, real-time line detection can be achieved [7].

In this paper, an efficient algorithm for line recognition is presented. The algorithm involves simple mathematical operations using integer arithmetic only, rather than floating point or expensive matrix operations.

2. LINE RECOGNITION ALGORITHM THEORY

The equation of a line is $y=mx+b$, where m is the slope of a line, and b is the y-intercept. The slope

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

for any two points (x_1,y_1), (x_2,y_2) that lie on the line. On the computer graphic devices such as screen, printer and plotter or any image file like BMP and JPEG, an image consists of pixels. The line that is drawn using grids is actually an approximation of the ideal/actual line. This line is formed by a set of adjacent straight lines that are close to each other using corner pixels.

In this paper, an algorithm to find the properties of a line using the properties of those actual straight line segments that form a line is presented.

The slope of an ideal line is as in (1). In the first octant where the slope is between 0 and 1, i.e.,

$$0 \le m \le 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

Substituting for m in (2) yields

$$0 \le \Delta y \le \Delta x \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

In the grid layout, all computations are done using integers. Even if you are using floating point calculations, once you need to know if the grid point is illuminated or not, rounding the value to an integer number is usually performed. So the ideal line is represented on the grid layout using a set of straight line segments. If the first and fourth octants of the plane are considered, these straight line segments are horizontal lines. If the line lies in the second or third octant, these straight lines are vertical lines. In this paper, our approach deals with the first octant, so we are considering horizontal lines. By this, the ideal line is formed using a number of straight lines. The graph of a line on the grid layout looks like stairs, as shown in Figure 1

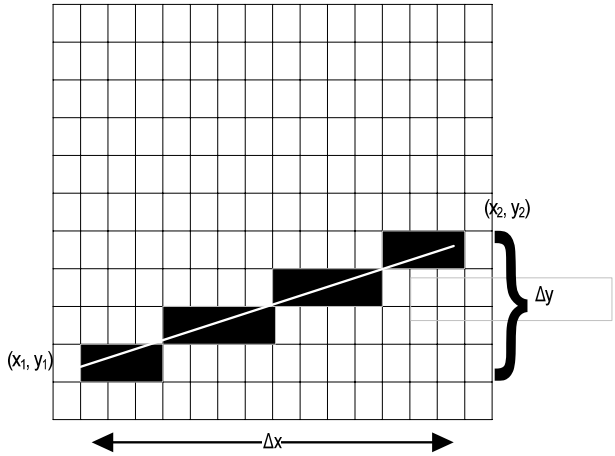

Figure 1. Ideal line drawn on a grid layout

Each increment in the y-direction is only one pixel, while the increment in the horizontal direction varies according to the end points of the line. The number of horizontal lines segments is Δy. The length of each horizontal line segment depends on Δy and Δx,

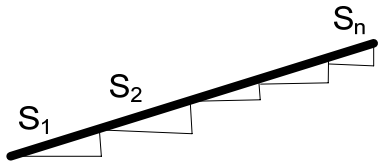

Figure 2. Horizontal line segments

The line that is drawn on a grid layout is formed by adjacent equal segments $\{S_1, S_2, \ldots, S_n\}$ if each line segment has the same slope as the ideal line. Figure 2 shows an ideal line with a number of horizontal line segments representing the approximation of this line. This means that the slope for each segment is equal to the slope of the whole line. The slope of each segment is

$$m = \frac{\Delta y'}{\Delta x'} = \frac{\Delta y}{\Delta x}$$

Since integer computations are performed, the slope is actually an approximation. The vertical increment is always one i.e. $\Delta y' = 1$ as it is a grid layout, Therefore $\Delta x' \cong \frac{\Delta x}{\Delta y}$

The ideal line passes at the middle of each vertical line segment, which has a length of one pixel as shown in Figure 3. This line passes closely to the middle of every vertical line segment ($\Delta y = 1$), which is

$$\Delta y'' \cong \frac{\Delta y'}{2}$$

The slope of the segment that passes at the middle of the vertical increment is

$$m'' = \frac{\Delta y''}{\Delta x''} = \frac{\Delta y'}{2\Delta x''} = \frac{\Delta y}{\Delta x}$$

This leads to

$$\Delta x'' = \frac{\Delta x}{2\Delta y} \quad \text{……. (4)}$$

The first and last horizontal segments lengths are equal to the value found in (4). The middle horizontal segments lengths have values close to the double of the first and last horizontal segments lengths i.e.

$$\Delta x' \cong \frac{\Delta x}{\Delta y} .$$

3. LINE RECOGNITION ALGORITHM

The algorithm is used to recognize a straight line in a graphic image. It works for the first octant of the plane, where the slope is between 0 and 1. Generalizing the algorithm for the whole plane is straight forward. The main advantage of the algorithm is its simplicity and robustness. The algorithm can detect any line that is continuous in the plane.

The data structures required for the algorithm are a matrix for the image, a bit matrix for visited pixels, the number of rows and columns of the matrices which are dependent on the size of the image.

A new data type is used to save the values of the line. It has four integer values for the starting and ending points. To save computational time, the length of line segment is saved as a member of this new data type. In order to keep track of all lines within the same image, a list of lines data structure is required.

The algorithm is shown in Figure 4. The algorithm loops through all pixels of the image in a row-major order. It checks if the pixel is part of a line and is not visited, then it starts a sequence of operations to find the line. These operations start by finding the first horizontal line segment and consider it as a temporary line. Then from the end of that segment in the next row, it checks for another horizontal line. If another horizontal line is found, the length of the new horizontal line is checked. Its length should be between the length of the first horizontal segment and its double length. If the conditions fail, that means there is no more horizontal segment to be part of the line. Upon completing that, the line should be added to the list of recognized lines.

```
Line_Recognition_Algorithm (Input: Matrix, Output: Lines-List)
// Matrix: matrix representation of the image.
// Lines-List: a list of recognized lines from  the image

begin
  for i=1 to image-height
      for j = 1 to image-width
        if ((Matrix[i][j] ==lineColor) && Not done[i][j] )

                  set  MoreSeg to  TRUE;
                 r = i
                 c= j
                  find a horizontal segment at (r, c)
                 save the horizontal segment parameters
                        into Temp-Line
                 set the length of the horizontal segment to len
                 set maxlen to double the length of segment

                  change the status of those pixels to visited
                while there are MoreSeg
                    if  direction is RIGHTDOWN
                          // direction of  the   first octant
                          find next horizontal segment at (r+1,
                              Previous segment  c+1)
                     end if
                    if ( length of the segment is between
                                      len and maxlen)
                          Change the end point of the Temp-Line to
                           be the endpoint of the new segment
                          Change the status of those pixels to visited
                    else
                          MoreSeg = FALSE
                    end if
                end while


              add temporary  line to the Lines-List
        endif
end
```

Figure 4. Line recognition Algorithm

EXAMPLES:

Here are three examples of the algorithm using BMP files. In first two images, there is one line in each, while in the third one, there are two lines.

In the first example, the algorithm scans the image row by row. At pixel (1,1), the algorithm finds an illuminated pixel, which represents the starting pixel of a horizontal line segment consisting of pixels (1,1) and (1,2). The length of this segment is 2 pixels. The next segment

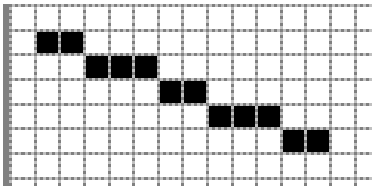

Figure 5. Example 1

starts at the corner of the previous segment i.e., r and c are incremented by 1. The new segment consists of pixels (2,3), (2,4) and (2,5) with length 3. The length is less than 4 which is the double of the length of the first segment. Then the line end points are (1, 1) and (2,5). It proceeds for the next horizontal segment which consists of pixels (3,6) and (3,7) with length 2, as the segments length is between 2 and 4. This line segment is also part of the line, so the line end points become (1,1) and (3,7). The same applies for the other two segments.

The second example is the same as the previous one, except that the lengths of the horizontal segments are 1 as shown in Figure 6. It shows that the number of maximum horizontal line segment's length is 2, which is the double of the length of the first horizontal segment.

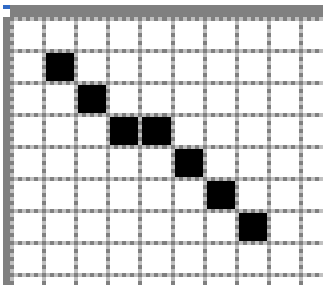

Figure 6. Example 2

In the third example, the first horizontal line segment's length is 4. This means that the other line segments' lengths must be between 4 and 8. The second horizontal line segment's length is 3, which is less than four, which means that this segment is not part of the line. Therefore the algorithm terminates at this line segment. So the first line end points are (1,1) and (1,4). By continuing the scanning of the rows, pixel (2,5) is reached. The first

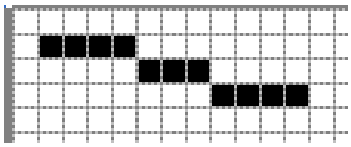

Figure 7. Example 3

horizontal line segment end points are (2,5) and (2,7) with length 3. The next horizontal segment is (3,8),(3,11) with length 4, which is between 3 and 6, so the second line end points become (2,5) and (3,11).

4. ALGORITHM ADVANTAGES AND CONCLUSIONS

All algorithms that are used for line recognition and detection in image processing involve very expensive operations like floating-point calculation and matrix operations. It is known that these operations are very time consuming and using them will definitely slow down these algorithms significantly. On the other hand, our algorithm uses only simple integer arithmetic. In addition to that, our algorithm involves the simplest forms of arithmetic operations namely, additions, comparisons and logical operations. There are no multiplications or divisions. Therefore, our algorithm has a great value in line recognition and image processing due to its simplicity and efficiency.

REFERENCES

[1] Guido M. Schuster and Aggelos K. Katsaggelos "Robust Line Detection Using A Weighted Mse Estimator", Image Processing, ICIP 2003. Proceedings. 2003 International Conference on Publication, Vol.1, pp. 293-296. 14-17 Sept. 2003

[2] R. Boyle and R. Thomas Computer Vision:A First Course, Blackwell Scientific Publications, 1988, Chap. 5.

[3] "Fast Line Detection Algorithms Based on Combinatorial Optimization", Marco Mattavelli1, Vincent Noel1, Edoardo Amaldi2,

[4] D.S. Guro, B. H. Shekar, P. Nagabhushan, “A simple and robust line detection algorithm based on small eigenvalue analysis”, Pattern Recognition Letters, Vol 25, 2004
[5] Yun-Seok Lee , Han-Suh Koo , Chang-Sung Jeong, A straight line detection using principal component analysis, Pattern Recognition Letters, v.27 n.14, pp.1744-1754, 15 October 2006
[6] D. Vernon Machine Vision, Prentice-Hall, 1991, Chap. 6
[7] Chan, T.S. Yip, R.K.K., Pattern Recognition, 1996., Proceedings of the 13th International Conference on , Volume: pp. 126-130, 25-29 Aug 1996