Ubiquitous Computing and Communication Journal
(ISSN 1992-8424)

LINE RECOGNITION IN IMAGE FILES USING INTEGER-BASED FORMAL APPROACH

Khalid Daghameen and Nabil Arman
Palestine Polytechnic University, Palestine
{dkhalid,narman}@ppu.edu

ABSTRACT

One of the main areas in image processing is line detection and recognition. Some algorithms for line detection were introduced. However, these algorithms involve expensive operations like floating-point arithmetic and matrix operations. In addition, the number of iterations of these expensive operations is high. Such operations increase the computational time significantly. In this paper, a line recognition algorithm, which involves integer arithmetic only, is introduced. The algorithm uses a formal approach to recognize a line within an image file.

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 involves the use of 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's 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. The algorithm uses digital line segment attributes. In discrete domain, a continuous line can be thought as sets of patterns combined in the quantized direction of edge pixels (i.e. 0°, 45°, 90°, 135°). These edge pixels have approximately equal number of pixels. Based on this property, the boundary convergence of discrete line is obtained. Based on the convergence property, a simple and efficient line detection algorithm is presented. Real-time line detection can be performed in a very simple and highly efficient manner [7].

An algorithm for recognizing a line in an image file was proposed in [8]. It was based on integer arithmetic, in which it decreases the processing time. The algorithm uses informal way to prove the correctness. Ours uses formal way to prove the correctness.

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

The algorithm is intended to be used for line detection. Circles, ellipses and other curves are not considered in this research since they are represented using mathematical models that are completely different form line mathematical representation.

2 LINE RECOGNITION ALGORITHM THEORY

The line equation between two points (x_1,y_1) and (x_2, y_2) is

$$y = m(x - x_1) + y_1 \quad \textbf{........................(1)}$$

where m is the slope and (x_1, y_1) is the first point of the line. It is a continuous equation for an ideal line, while in digital images, the situation is different, since x and y are both discrete values. This means that x takes a set of whole numbers between x_1 and x_2, and the same applies for y, i.e.

$$x \in \{x_0, x_1, x_2, \ldots., x_n\}$$

$$y \in \{y_0, y_1, y_2, \ldots., y_k\}$$

In digital form

$$x[n] = [x[0], y[1], y[2], \ldots, x[n]]$$

$$y[k] = [y[0], y[1], y[2], \ldots., y[k]]$$

where $x[0]=x_1$
and $y[0]=y_1$
as for $x[n]=x_2$ and $y[k]=y_2$

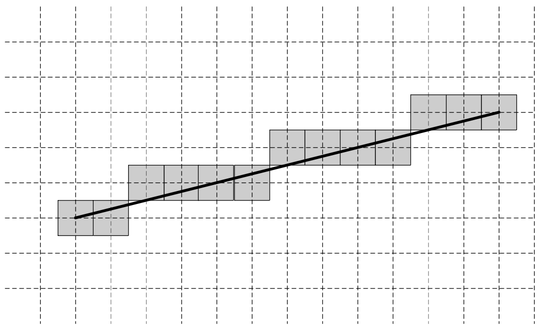

Figure 2: Line approximation on a grid layout

The difference between any two consecutive values is 1, i.e. $x[j+1]$-$x[j]$ $=1$ and $y[i+1]$-$y[i]$ $=1$. i.e. in figure 2, the points that are part of a line are (x_i, y_k), (x_i+1, y_k+1), (x_i+2, y_k+1), (x_i+3, y_k+1), (x_i+4, y_k+1), (x_i+5, y_k+2). Such point illustrates that above fact.

The difference between two consecutive x values is 1, while in y-direction the difference is m. Using

$$y = m(x - x_1) + y_1 \quad \textbf{........................(1)}$$

$$y[2] - y_1 = m(x[2] - x_1)$$

$$y[1] = 1m + y_1$$

$$y[2] = 2m + y_1$$

Therefore, we can say that

$$y[n] = mn + y_1$$

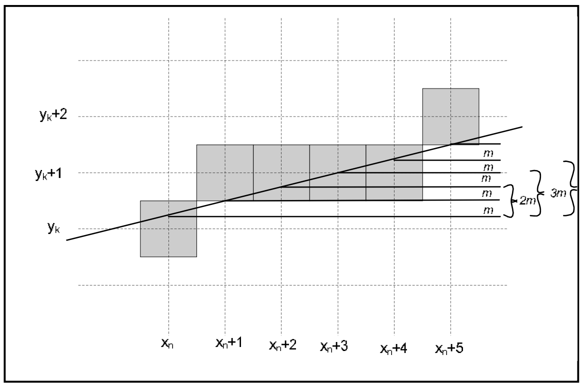

Figure 1: Relation between the increments in both directions

y-values are rounded to integer values (whole numbers) as in the following definition of a function:

$$y_k = \begin{cases} \lceil y_k \rceil & \lfloor y_k \rfloor + 0.5 \le y_k \prec \lceil y_k \rceil \\ \lfloor y_k \rfloor & \lfloor y_k \rfloor \prec y_k \le \lfloor y_k \rfloor + 0.5 \end{cases}$$

For the first horizontal line segment. The difference between the y's is 0.5, we need to find the number of increments of x's to reach the first value of y that is different from the current value. Using equation 2, we can find that.

$$y_k - y_1 = \frac{1}{2} = nm + y_1 - y_1$$

$$n = \frac{1}{2m} = \frac{\Delta x}{2\Delta y}$$

For any two points in the middle, the difference between any two consecutive y values is 1. So using the same equation above, the number of increments in the x-direction can be determined.

$$y_{k+1} - y_k = 1 = n_2 m + y_1 - (n_1 m + y_1)$$

$$n_2 m - n_1 m = 1$$

$$n_2 - n_1 = \frac{1}{m} = \frac{\Delta x}{\Delta y}$$

so, the number of x increments, which is the length of the horizontal line segment is equal to the reciprocal of the line slope for all the middle segments, while the outer horizontal segments is half in length for those in the middle line segments.

It is clear that every thing here is an approximation. So for the middle segments, the difference in the vertical displacement is not exactly one. It is less than one at most by m. This means, it is not less than 1-m , and it is not more than 1.

Thus,

$$y_{k+1} - y_k > 1 - m$$

$$n_2 m + y_1 - n_1 m - y_1 > 1 - m$$

$$n_2 - n_1 > \frac{1-m}{m}$$

$$n_2 - n_1 > \frac{\Delta x}{\Delta y} - 1$$

From this, it is concluded that, the middle horizontal segments length value is between the reciprocal of the slope and the length of the first horizontal line segment.

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 3. 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.

The algorithm can be extended to handle lines of different thicknesses by testing pixels adjacent to the main pixels comprising the main line. The level of thickness determines number of adjacent pixels to be tested. Here are three examples of the algorithm using BMP files. In the first two images, there is one line while in the third one, there are two 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 3: Line recognition Algorithm

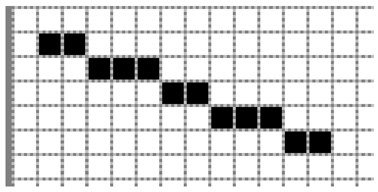

Figure 4: One line in a BMP file

In the first example, the algorithm scans the image row by row. At pixel (1,1) as shown in Figure 4, 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 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 by one, 4 is the double of the length 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 equal to the length of the first segment. 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 5. 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.

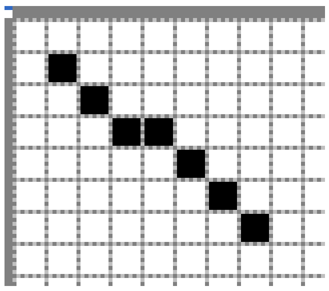

Figure 5: One line in a BMP file

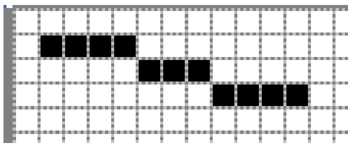

Figure 6: More than one line in a BMP file

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 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).

Clearly, the algorithm outperforms other algorithms for line detection in image files since it uses integer arithmetic in its operations. As mentioned before, other approaches and algorithms use more expensive operations.

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.

5 REFERENCES

[1] G. M. Schuster and A. K. Katsaggelos: Robust Line Detection Using A Weighted Mse Estimator**,** Image Processing, ICIP 2003. Proceedings of 2003 International Conference on Publication, 14-17 Sept., Vol. 1, pp. 293-296, (2003)

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

[3] M. Mattavelli1, V. Noel1, and E. Amaldi: Fast Line Detection Algorithms Based on Combinatorial Optimization, Lecture Notes In Computer Science; Vol. 2059, 2001.

[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] Y. Lee , H. Koo , and C. Jeong: A straight line detection using principal component analysis, Pattern Recognition Letters, Vol. 27 No. 14, pp. 1744-1754, (2006).

[6] D. Vernon: Machine Vision, Prentice-Hall, Chap. 6, (1991).

[7] T. S. Chan, R. K. Yip: Pattern Recognition, Proceedings of the 13th International Conference on Pattern Recognition, 25-29 Aug, pp. 126-130, (1996).

[8] K. Daghameen, and N. Arman, An Efficient Algorithm For Line Recognition Based On Integer Arithmetic, Proceedings of 2^{nd} Palestinian International Conference on Computer and Information Technology, 1-3 Sep., pp-20-24, (2007).