Implementation of Distance Transformation in the Processing Language

Rama Prasada Reddy Peddireddy
Department of Computer Science,
University of Colorado, Colorado Springs,
CO, USA
rpeddire@uccs.edu

Sudhanshu Kumar Semwal
Department of Computer Science,
University of Colorado, Colorado Springs,
CO, USA
ssemwal@uccs.edu

***Abstract*—In this paper, we describe three different approaches for determining or finding a distance map for a binary image. The algorithms that solve such problems are known as Distance Transforms. These algorithms that solve such problems are known as Distance Transformations. These algorithms operate on binary images but can be extended to receive any type of digital image if a conversion algorithm that converts a digital image into a binary digital image is executed prior to executing the Distance Transform algorithm. Therefore, we also examine how to transform any regular digital image into a binary image, that is, into a black and white image. A Distance Transformation algorithm operates on a binary image consisting of featured pixels and non-featured pixels. It outputs a distance map or distance matrix where each cell matches a pixel of the input image and contains a value indicating the distance to the nearest featured pixel. Distance Transforms represent a natural way to blur feature locations geometrically and they allow other image effects like skeletonizing, image matching, object recognition, path planning and navigation. Five test cases are presented and the execution times of the three techniques are compared.**

Keywords—City-block; Euclidian distance transformations; image processing; Processing language

I. INTRODUCTION

Our objective is to implement distance transformation [1], [2], [4]-[7] using processing programming language using a binary image as an input. To understand distance transformations, first we need to understand the fundamentals of image processing such as image types, pixels in an image, pixel values and pixel connectivity. Pixel connectivity is a relation between the neighboring pixels. There are different types of pixel connectivity. For two-dimensional images we explained three basic pixel connectivity and they are explained below. In four-connected pixel connectivity, pixels are connected by their sides. This means that a pair of pixels are connected horizontally or vertically [2], [4], [10]. Every pixel that has the coordinates $(x \pm 1, y)$ or $(x, y \pm 1)$ is connected to the pixel at (x, y). In diagonal-connected pixel connectivity, pixels are connected by their corners. This means that a pair of pixels is connected diagonally [17]. Every pixel that has the coordinates $(x-1, y\pm1)$ or $(x + 1, y \pm 1)$ is connected to the pixel at (x, y). This connectivity is union of both four-connected pixel connectivity and diagonal-connected pixel connectivity. Every pixel that has the coordinates $(x \pm 1, y \pm 1)$ is connected to the pixel at (x, y) [2], [4], [10].

The Processing language is an open source software environment and a programming language for artists who want to program images, animation and sound. The objective behind the creation of processing is to guide fundamentals of computer programming within a visual context and to serve the programmers as a software sketchbook and professional production tool. Processing is developed by artists and designers to serve artists as a tool to design sketches with in the same domain. Processing is consisting of all the principles, structures and concepts like other programming languages [11]. The main advantages of the Processing software are free to download by anyone and runs on the Mac, Windows, and Linux platforms and it's developing environments is very flexible as well as user friendly to code. The processing programming language is a text programming language and as mentioned above its main aim is to generate new images as well as alterations for existing images [11].

II. DISTANCE TRANSFORMATIONS

Distance transformations are well known as a technique to compute distances from the featured pixel to non-featured pixels. There are various distance transformations proposed and among all these there are three distance transformations which have been proposed for two-dimensions [2], [3].

A distance transformation is an operation that converts a binary picture, consisting of feature and non-feature elements, to a picture where each element has a value that approximates the distance to the nearest feature element [3]. The distance values of these featured and non-featured pixels are computed based on its neighbors. Computing the distance between a non-feature pixel to a feature pixel is essentially a global operation. In case of large images these operations are computationally costly unless the images are very small [2].

Consider a binary image which consists of a featured pixel and non-featured pixels. These featured pixels can be points, edges or objects and non-featured pixels are unoccupied (blanks). The challenge is to calculate the distance from non-featured pixel to nearest featured pixel. This distance calculation is computationally costly to begin with every non-featured pixel and to scan the image until we find a featured pixel [2]. In this paper, we particularly concentrated on implementing the distance transformation algorithms which are proposed by Borgefors. There are several distance metrics to calculate the distance between non-featured pixel and featured pixel. In two-dimensions there are proposed three main

distance transformations and they are known as city block, chessboard and Euclidean distances [1]. These distance transformations can be implemented using both sequential and parallel algorithms [2]. A brief explanation about these techniques is in the next section. The same basic idea is involved in all three distance transformations and it is approximations of the global Euclidean distance which then can be computed using local operations.

Several issues arise when dealing with distance transform algorithms. One of these issues and perhaps the most significant of all is accuracy. Some distance transform algorithms produce results with minimal errors; others are theoretically proven to be error free. From a scientific point of view it is important to be able to validate the implementation of the algorithm. A second important issue relates to the computational time that it takes to provide an output. It is relatively straightforward to develop an exhaustive method that requires a great deal of processing time and computes the distance map previously mentioned [5]. In general distance transform algorithms exhibit varying degrees of accuracy of the result, computational complexity, hardware requirements (such as parallel processors) and conceptual complexity of the algorithms themselves.

Distance transform algorithms are relying on the idea of propagating distances between pixels. Calculating the distance map can have a high computational time cost therefore the strategy applied by distance transform algorithms is to approximate the global or real distances by propagating the local distances, that is, the distance between neighbor pixels [2]. The propagation can be accomplished sequentially or in parallel. The algorithms presented in this document were developed using a sequential approach.

In distance transform algorithms, the distance map has the same dimension as the input image and it's initializing as follows [3], [5]:

If x = [i, j] is a featured pixel then dist_map [i, j] = 0

else dist_map [i, j] = Infinity

Note that every featured pixel will have a value of 0 in the distance map and every non-featured pixel a value of Infinity.

Setting these initial values this way will prove to be necessary when propagating local distances and calculating the rest of the distance map.

Thus, distance measure has been used for the calculation of the distance but no description of such distance measure has been provided. The distance measures that will be applied in the algorithm described throughout this paper are: the Manhattan or City Block Distance, the Chessboard Distance and the Euclidean Distance.

After the initialization stage a distance transform algorithm executes two stages [3] over the image, one is called forward propagation and the other backward propagation. In these stages, the algorithm typically uses a mask that is distance measure dependent and is divided into two masks, one for the forward propagation known as forward mask and another for the backward propagation known as backward mask (Fig. 1 and 2). The forward propagation consists of a sweep of the image from left to right and top to bottom applying the forward mask every moment and updating the value of cells in the distance map as follows:

Dist_map[j] ←min(Dist_map[j], Dist_map[j-1] +1)

In the forward propagation, the distance map is updated by considering the value of the previous cell adding the neighbor distance (adjacent cells are always at distance 1) and the distance value of the current cell. Note that in the last formula refers to the value of the pixel at position

$$X = (i, j).$$

In the backward propagation, you sweep the image from right to left and bottom to top applying the backward mask and updating values according to the next formula

Dist_map[j] ←min(Dist_map[j], Dist_map[j+1] +1)

As one can notice the formula both formulas are similar, the only transcendent change is the direction of the update. Assuming a distance transform algorithm and a Manhattan distance measure (explained shortly) which receive an input image with a single featured pixel in the middle the results obtained from the forward and backward passes would be the ones illustrated in the following figure.

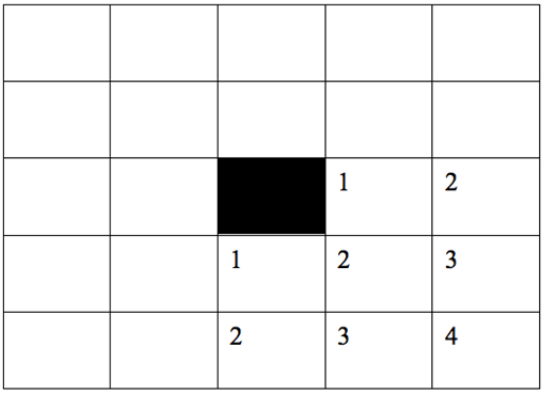

Fig. 1. Forward mask propagation.

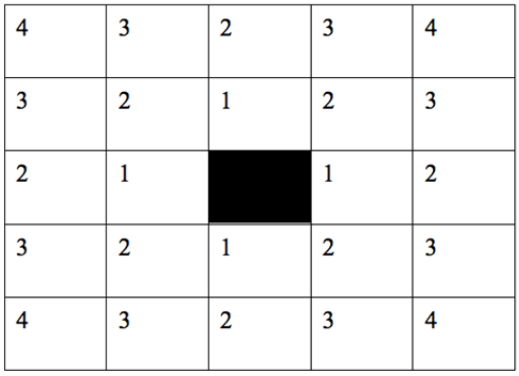

Fig. 2. Backward mask propagation.

III. Distance Transormation Types

After getting a good idea on distance transformation three distance transformation techniques are explained next. These

are Chessboard, City-block and Euclidean distance transformations based on their distance metrics.

A. Chessboad Distance Transformation

The chessboard distance metric measures the path between the pixels based on 8-connected neighborhood (Fig. 3(a)). Pixels are connected if their edges or corners touch. This means that if two adjoining pixels are on, they are part of the same object, regardless of whether they are connected along the horizontal, vertical, or diagonal direction [2]. Pixels whose edges or corners are one unit apart. On the other hand, we can say this metric assumes that you can make moves on the pixel grid as if you are a king making moves in chess game. Suppose that two points (x1, y1), (x2, y2) are given. Fig. 3(c) show the chessboard distance transformation technique implemented for a simple image shown in 3b.

The chessboard distance is then defined as: D = Max {|x1 - x2|, |y1- y2|.

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Fig. 3. (a): Chessboard distance transformation with feature pixel at the center.

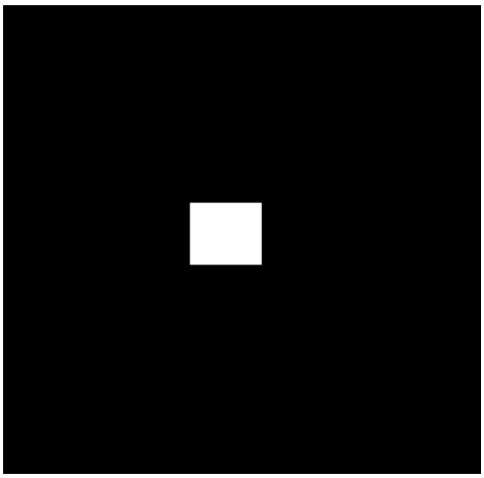

Fig. 3. (b): Input binary image to our program.

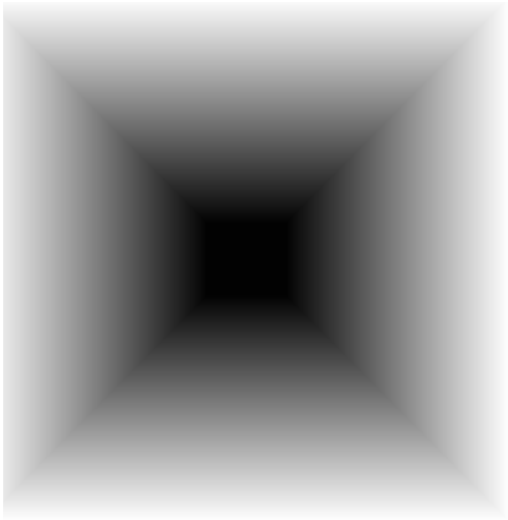

Fig. 3. (c): The chessboard distance transformation using the Processing language.

B. City Block Distancet Transformation

The city block distance metric measures (Fig. 4(a)) the path between the pixels based on a four-connected neighborhood. Pixels whose edges touch are one of unit apart, pixels diagonally touching are two units apart. It is also known as the Manhattan distance. This metric assumes that it is going from one pixel to other. It is only possible to travel directly along pixel gridlines. In this technique, diagonal moves are not allowed [2], [3].

Suppose that two points (x1, y1), (x2, y2) are given.

Then the City block distance is then defined as: D =|x1-x2|+|y1-y2|. Fig. 4(b) shows a simple image and its city block distance transformation (Fig. 4(c)).

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

Fig. 4. (a): City-block distance transformation with feature pixel at the center.

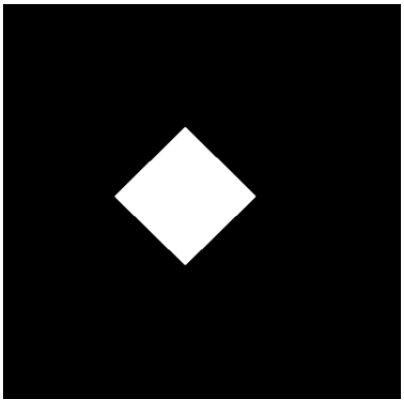

Fig. 4. (b): Original image to the processing program.

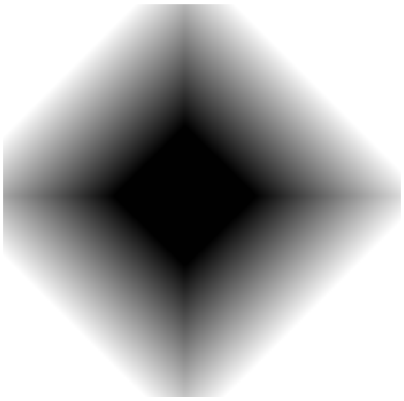

Fig. 4. (c): City-block distance transformation of the image in Fig. 4(b).

C. Euclidean Distance Transformation

The Euclidean distance transformation is a familiar straight line distance between two points [2], [7] (Fig. 5(a)). Suppose that two points (x1, y1), (x2, y2) are given. Then the chessboard distance is then defined as:

$$D = \sqrt{(x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2)}.$$

Fig. 5(b) and 5(c) show a simple image and its Euclidean distance transformation.

1.4142	1	1.4142	2.2361	3.1623
1	0	1	2	2.2361
1.4142	1	1.4142	1	1.4142
2.2361	2	1	0	1
3.1623	2.2361	1.4142	1	1.4142

Fig. 5. (a): Euclidean distance transformation with two feature pixels at (2,2) and (4,4).

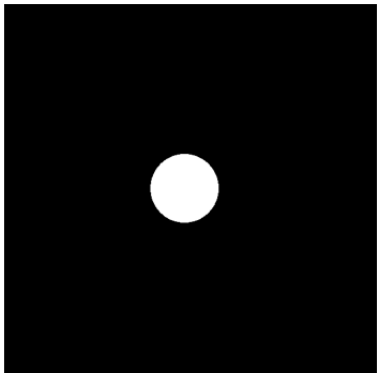

Fig. 5. (b): Input image to the Euclidean distance transformation program.

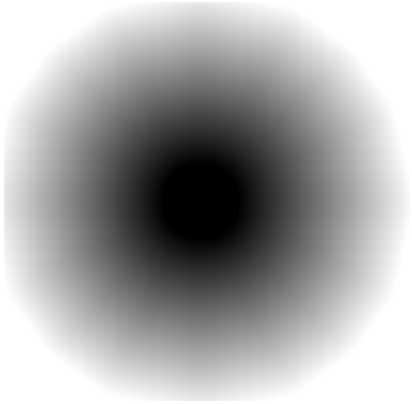

Fig. 5. (c): Euclidean distance transformation implemented using processing.

IV. IMPLEMENTATION

In converting a color or digital image into a binary image we use the r, g, b value, and the following equation:

$0.3*r + 0.59*g + 0.11*b > 127$

With 127 being used as the threshold [8], [9] for deciding whether a pixel is colored, black or white.

A. Minimum Distance Calculation

Next, we are going to explain how are we calculating the minimum distance from non-feature pixel to the nearest featured pixel with the mask. In the first part of the pseudo code we are focusing backward mask propagation and in the later part about forward mask propagation. The both forward and backward propagations are very similar except in the direction of mask propagation. To calculate this minimum distance value, we need to assign the pixel values as Infinity for non-featured pixel and zero for featured pixel. Based on mask size here we are calculating a bandwidth which is a side width of the mask for both forward mask and backward mask propagations. We are using for loops here to find out various pixels located on the images. Current state parameter gives the distance value of the pixel once we identify the location of the pixel. The co-ordinates of the pixels are useful to find out distance based on distance transformation technique [2], [3].

B. Main Transformative Process

In this section, we are discussing the distance transformation of a binary image using forward scan and backward scan for sequential algorithm [2]. To process output

image, we must initialize a process for creating an empty image along with a distance matrix with an empty data metrics. The next step is to get a pixel value at the position (i, j) which is a co-ordinate of image. The distance matrix identifies these positions of the pixel with image height and width. To perform distance transformation, we need to find the pixel's state at the position and we can achieve this by comparing the temporary distance transformation array which is a distance matrix. In case of sequential algorithms this process takes place twice. One is forward scan and other one is backward scan. In the forward scan, it scans the image from left to right and top to bottom to find out the distance based on nearest featured pixel and other hand the backward scan propagates from right to left as well as bottom to top as explained in [3].

C. Output Image

Using the distance matrix, distance values are mapped to grey scale and displayed.

V. Results

We have implemented five different tests. We provide the results of each distance transformation types below. Fig. 6 shows the result of a black and white hand-drawn image, with chessboard, city-block and Euclidean distance transformation applied to the same image. Chessboard, city block and Euclidean distance transformations are also shown for a variety of black and white images in Fig. 7 and 8 as well. Next we took some color images in Fig. 9 from Google images, and applied our algorithm to their equivalent black and white images after thresholding [7]-[11] in Fig. 10 and 11. Fig. 12 shows a chart of execution times for the three techniques for all five images which we tested in our implementation. They clearly show that for the same image, chessboard is the least expensive and Euclidean distance transformation is the most expensive processing technique. Execution time for the city-block distance transformation technique is in between the chessboard and the Euclidean distance transformation times.

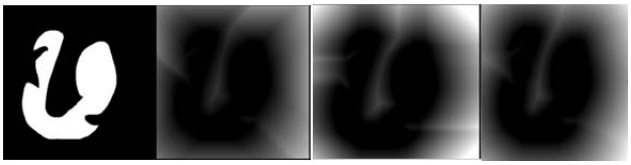

Fig. 6. Test 1: Left original, chessboard, city-block, Euclidean (right).

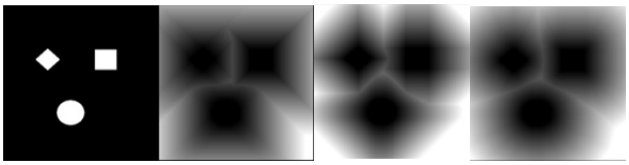

Fig. 7. Test 2: Left original, chessboard, city-block, Euclidean (right).

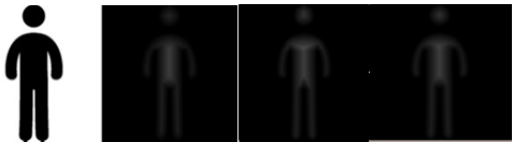

Fig. 8. Test 3: Left original, chessboard, city-block, Euclidean (right).

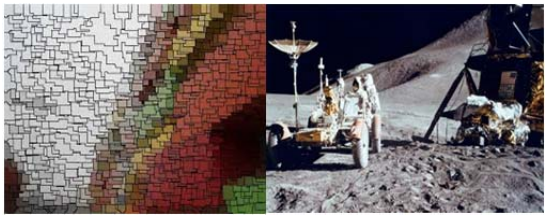

Fig. 9. Tests 4 and 5 images.

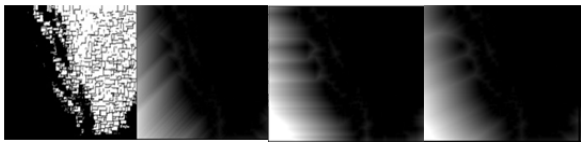

Fig. 10. Test 4: Left grey scale, chessboard, city-block, Euclidean (right).

Fig. 11. Test 5: Left grey scale, chessboard, city-block, Euclidean (right).

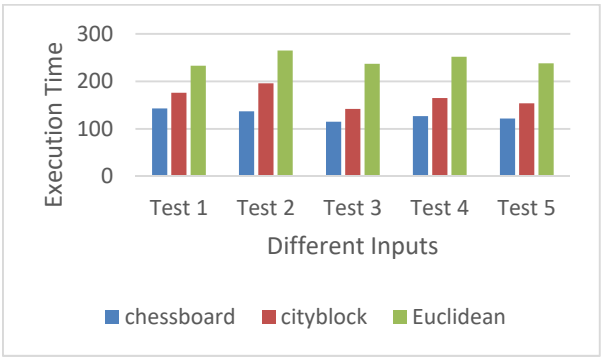

Fig. 12. Comparison of the three techniques.

VI. Conclusions and Future Work

We described three distance transform algorithms and some of the different distance measures that are used in these algorithms. Namely, we detailed the City Block, Chessboard and Euclidean distances; the first two being the most efficient but less accurate, the last one being the most accurate but less efficient because of its floating-point calculations.

During the experimentation and comparison phase which implemented different images we could see how distance transforms using the Euclidean distance provided the most reliable and close to reality results whereas city block and chessboard in many cases introduced noise in the resulting image because of the way the distance map is calculated and the discretization errors committed in the process.

As future work, we recommend implementing new methods that could provide better results and try to incorporate to these new algorithms mask of size greater values, and corresponding 3D implementations. Distance transformations are the mainstay for several image processing techniques, and have been well-studied in the past. We have implemented these techniques in the Processing language. Our future goal is to

look into extending the distance transformation techniques for novel medical applications.

ACKNOWLEDGMENT

All binary images were created in Paint Program. Test 3 is from free pic. Tests 4 and 5 borrowed from freely available Google images.

REFERENCES

[1] G. Borgefors, “Distance transformations in arbitrary dimensions”. Comput. Vision, Graphics Image Process. 27, 1984, pp. 321-345.

[2] G. Borgefors, “Distance transformations in digital images”. Computer Vision, Graphics, Image Processing, 34, 1986, pp. 344-371.

[3] S. K. Semwal and H. Kvarnstrom. “Directed Safe Zones and the Dual Extend Algorithms for Efficient Grid Tracing during Ray Tracing”. In Proceedings of Graphics Interface '97, pp. 76–87, May 1997.

[4] G Borgefors, "Hierarchical chamfer matching: a parametric edge matching algorithm", *IEEE Trans PAMI*, vol. 10, no. 6, pp. 849-865, 1988.

[5] Grevera, G.J, “Distance transform algorithms and their implementation and evaluation. In: Deformable Models”, pp 33–60, 2007.

[6] E. L. van den Broek, T. E. Schouten, "Distance transforms: Academics versus industry", *Recent Patents Comput. Sci.*, vol. 4, no. 1, pp. 1-15, 2011.

[7] H. Breu, J. Gil, D. Kirkpatrick, M. Werman, "Linear time Euclidean Distance Transform Algorithms", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, pp. 529-533, 1995

[8] Morse, Bryan S., "Lecture 4: Thresholding," Brigham Young University, 1998-2000, pp. 1-5.

[9] P. K. Sahoo, S. Soltani, A. K. C. Wong, "A survey of thresholding techniques", *Comput. Vis. Graph. Image Process.*, vol. 41, pp. 233-260, 1988.

[10] Bryan S. Morse, “Lecture 2- Image Processing Review, Neighbors, Connected Components, and Distance “, Brigham Young University, 1998-2000, pp. 1-7.

[11] Casey Reas, Ben Fry, John Maeda, Processing: A Programming Handbook for Visual Designers and Artists, The MIT Press, 2007.