Distance Transform

Project: Marker-based Localization

Niharika Jayanthi, Dheeraj Kamath
Mentor: Sanam Shakya

Goal

- To understand what is distance transformation.
- Learn about chessboard, Euclidean and city-blocks distance metrics.
- Learn how to code for distance transformation in python.

Theory

Working Principle

Introduction

Distance transform is an operator that transforms the values of pixels in an image according to their distance from the boundary of the object. The farther a pixel is from the image boundary, the higher a value it gets. This would result in a transformed image where the area near and within the boundary will be dark whereas the area near the center of the image will be white.

To put it in different words, a pixel in an image can have a value equal to the number of erosions required to be performed to make it disappear.

Distance transform operations can be classified into several types, depending upon the way the distance of a pixel from the boundary has been calculated (Distance metrics). Below are the three common distance metrics-

- **Chessboard Distance**

This is analogous to a King on the chessboard, which can move horizontally, vertically and diagonally. All moves are considered equal to each other. Hence, distance between two points (x1, y1) and (x2, y2) is calculated as-

```
D = max(|x2 - x1|,|y2 - y1|)
```

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

Figure 1: Chessboard Distance

In the above image, each cell represents a pixel. As you can see, the value of the pixel increases as it moves away from the boundary. This distance is also known as Chebyshev distance.

- **Euclidean Distance**

This is equal to the length of the line segment formed by joining two points. The following formula is used to calculate Euclidean distance between two points A(x1, y1) and B(x2, y2)-

```
D = squareRoot( (x2 - x1)^2 + (y2 - y1)^2 )
```

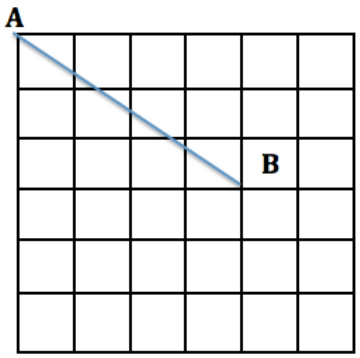

Figure 2: Euclidean Distance

It is evident from the above example that the direct, straight distance between the two points is calculated.

- **City-blocks Distance**

Also known as Taxicab distance or Manhattan distance, this method measures distance by traveling along grid lines. Therefore, diagonal moves are not possible. The formula for distance between two points A(x1, y1) and B(x2, y2) is-

```
D = |x2 - x1| + |y2 - y1|
```

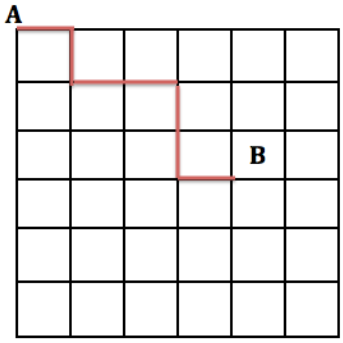

Figure 3: City Blocks distance

In the above example, you can see how path can either move horizontally or vertically only. The total distance would therefore be the sum of the differences in x and y coordinates respectively.

Applications

- It is used to generate skeletons of images. Skeletons depict the connectivity, length and width of an image.
- Distance transform can be applied in robotics for navigation and pathfinding.
- Font smoothing and rendering also use distance transform.
- 3D modelling uses signed distance fields.

Code

The main function used for distance transformation is-

```
cv2.distanceTransform(src, distanceType, maskSize)
```

Here,

```
src:          8-bit binary source image
distanceType: Type of distance. It can be CV_DIST_L1, CV_DIST_L2 or CV_DIST_C
maskSize:     Size of distance transform mask
```

Explaining the code

- Let us first import the required packages.

```
import cv2
import matplotlib.pyplot as plt
```

cv2 package is required as it contains the distance transform function. Matplotlib is needed to plot our images on the window.

- Next, we must read the image and convert it to a grayscale image.

```
img = cv2.imread('example.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- We need a binary image as input for distance transform. So gray will be converted into a binary image using thresholding.

```
ret, th = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
```

- We can now perform distance transformation on the thresholded image.

```
distance = cv2.distanceTransform(th, cv2.cv.CV_DIST_L2, 5)
```

- Now, let us display the grayscale and distance transformed image on a window. First, we create a new window.

```
plt.figure(0)
```

- The grayscale image will be plotted on the left side of the window with the following code-

```
plt.subplot(121)
plot = plt.imshow(gray)
plot.set_cmap('bone')
```

By plt.subplot(121), we create a 1 row by 2 columns grid and position the subplot in the first subplot.

- Now, we plot the second image on the right side of the window.

```
plt.subplot(122)
plot = plt.imshow(distance)
plot.set_cmap('bone')
```

- Finally, we display our window using-

```
plt.show()
```

Consider the following image with different shapes-

Figure 4: EXAMPLE

On performing distance transformation, it will look like this-

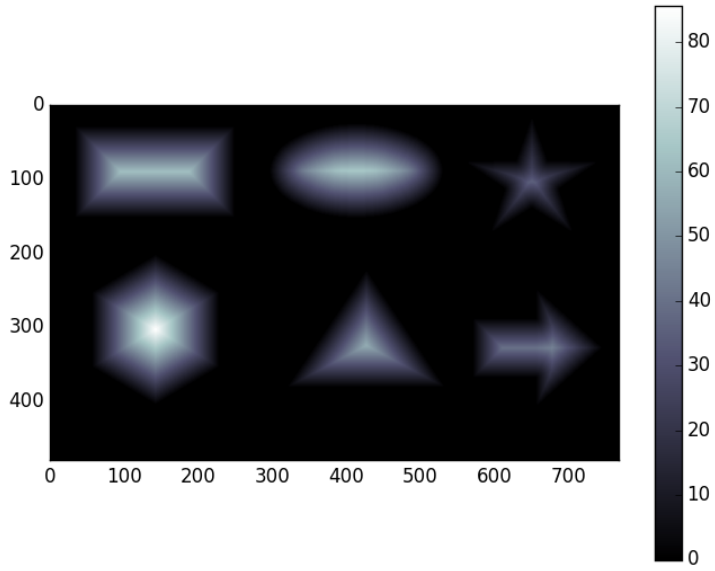

Figure 5: Result

The complete code can be found here.

References

- http://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/metric.htm
- http://en.wikipedia.org/wiki/Distance_transform
- http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html#distancetransform
- http://matplotlib.org/api/pyplot_api.html