Image Understanding – Assignment #4

Matching

1. Part I-Distance Transform

1.1 Organization of the code

The main function is DistanceMap.m. The information of this function is shown in Table 1.1. This function uses two-pass algorithm to calculate the distance map.

Table 1. 1 Description of DistanceMap Function

Name	DistanceMap		
Description	Calculate the distance map of the input image by two-pass algorithm		
	Name	Type	Description
Input	*I*	matrix	the matrix storing the input image
Output	*J*	matrix	the distance map of the input image

1.2 Result of binary image of cells

The distance map of binary image of cells using DistanceMap.m is shown in Fig 1.1.

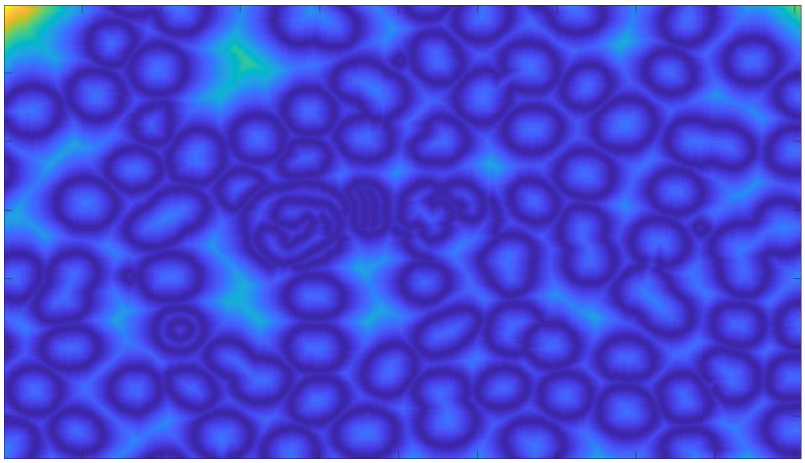


Fig 1.1 Result of distance map of cells image with range scaling

2. Part II- Hausdorff Matching

2.1 Organization of the code

The main function is HausdorffMatching.m. The information of this function is shown in Table 2.1.

Table 2.1 Description of HausdorffMatching Function

Name	HausdorffMatching		
Description	match the given template in the given image by using Hausdoff method		
	Name	Type	Description
Input	*I*	matrix	the matrix storing the input binary image
	T	matrix	the matrix storing the input binary template
	distance	int	optional parameter, set the width of dilated map
	coefficient	double	optional parameter, set the threshold coefficient

Also, there are several helper functions which are called by the main function. Below is the description of those functions.

Table 2.2 Description of Dilate Function

Name	Dilate		
Description	Dilate the input image by given parameter of distance		
Input	Name	Type	Description
	I	matrix	the matrix storing the input binary image
	distance	int	optional parameter, set the width of dilated map
Output	*J*	matrix	the matrix storing the dilated binary image

The function to generate the dilated image of the input image by using the "DistanceMap" implemented in Part I, then threshold the result by given parameter "*distance*".

Table 2.3 Description of LocalMaximaAndThreshold Function

Name	LocalMaximaAndThreshold		
Description	Threshold the image and find the local maxima within the given spec		
Input	Name	Type	Description
	I	matrix	the matrix storing the input image
	threshold	int	parameter of threshold
	length	int	parameter of length
	width	int	parameter of width
Output	*J*	matrix	the matrix storing the coordinate of the local maxima

Table 2.4 Description of DrawRectangle Function

Name	DrawRectangle		
Description	Draw rectangles on the original image based on input coordinate matrix and rectangle size		
Input	Name	Type	Description
	I	matrix	the matrix storing the matrix of centers of rectangles
	J	matrix	the matrix storing the original image
	length	int	parameter of length of the rectangle
	width	int	parameter of width of the rectangle

2.2 Find the local maxima

For thresholding and finding local maxima, several steps are used.

Step 1: Generate the threshold value. A given percentile value is used to calculate the threshold. The max value of the correlation matrix is multiplied by the percentile coefficient to produce the threshold for each matching.

Step 2: Loop the whole image to find the pixels where the value is above the threshold.

Step 3: All the neighbors of the pixels mentioned above are searched to see whether the pixel has the largest value among all the neighbors. Those local maxima are stored in a matrix for further operation.

Step 4: Compare each pair of local maxima within the given spec, the bigger one will be preserved, and smaller one will be thrown. This operation is used to find the maxima within the given spec size so that within a matching object, there will be only one matching pixel as the center of the matching object.

2.3 Result of Hausdorff Matching

A template image of cell is cropped from the original image to test the Hausdorff Matching function.

Test code:

```
I = imread('cells.png');                    %read the input image and store it in matrix I
T = im2double(I(181:216,87:126));           %crop a cell from the input image as the template image
HausdorffMatching(I,T,2,0.99);              %match the template in input image with given parameters
```

The image of template is shown in Fig. 2.1.

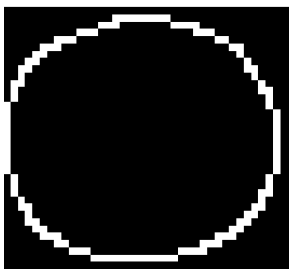


Fig 2.1 Template image of a cell

Results of Hausdorff Matching with different input parameters are shown as below.

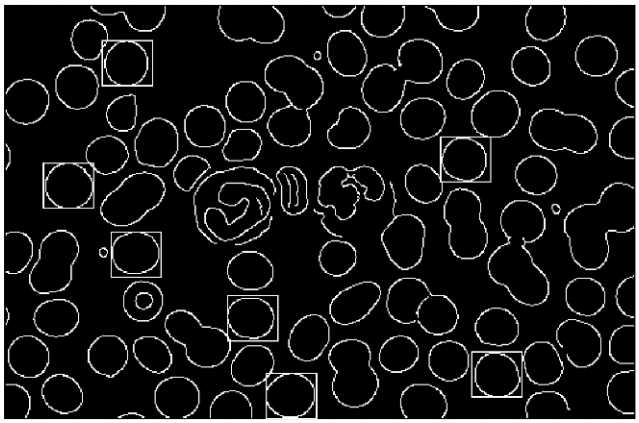
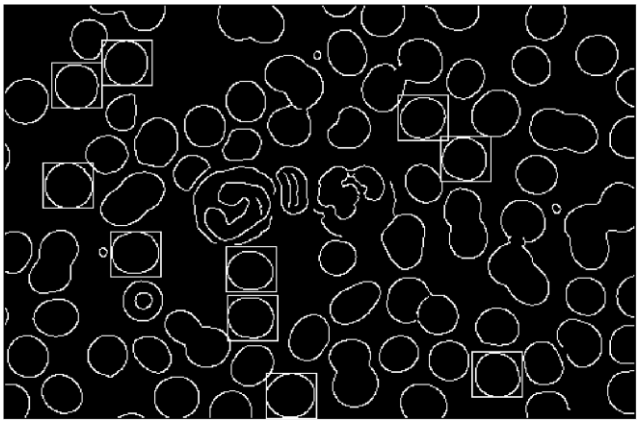


Fig 2.2 Matching result with distance = 3, coefficient = 0.9999



. Fig 2.3 Matching result with distance = 3, coefficient = 0.95

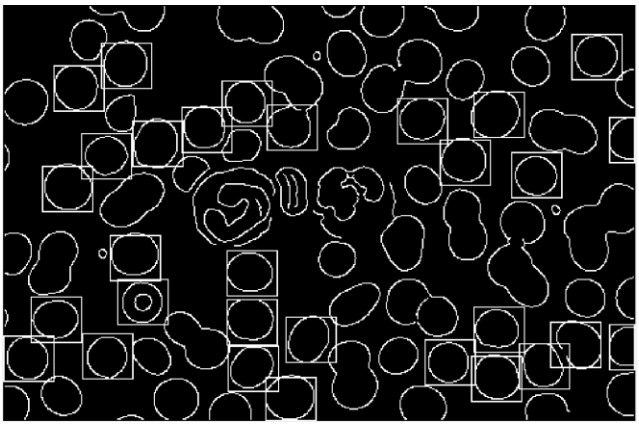
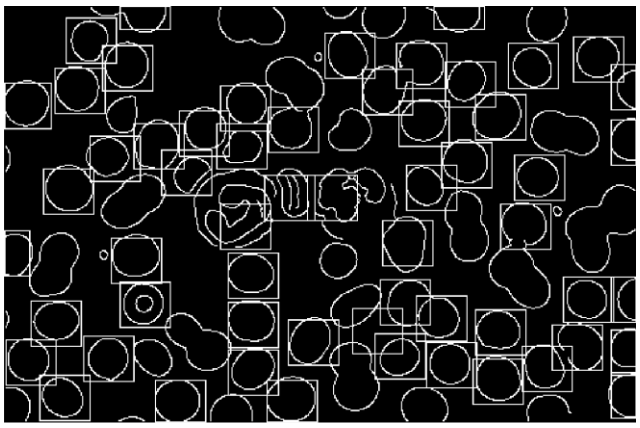


Fig 2.4 Matching result with distance = 6, coefficient = 0.9999



. Fig 2.5 Matching result with distance = 3, coefficient = 0.95

When distance increases, the function can detect matchings of different sizes. And when coefficient decreases, the function can detect more matchings. However, if the coefficient is small enough, some wrong matching will be detected, such as the result shown in Fig.2.5.

3. Part III- Matching Over Objects of Different Sizes

3.1 Organization of the code

The main function is CoinMatching.m. The information of this function is shown in Table 3.1.

Table 3.1 Description of CoinMatching Function

Name	CoinMatching		
Description	Match the coins in the input image with different sizes		
	Name	Type	Description
Input	*I*	matrix	the matrix storing the input binary image

Also, there are several additional helper functions which are called by the main function. Below is the description of those functions.

Table 3.2 Description of LocalMaximaAndThreshold_DifferentSizes Function

Name	LocalMaximaAndThreshold_DifferentSizes		
Description	Threshold the image and find the local maxima within the given spec		
Input	Name	Type	Description
	I	matrix	the matrix storing the input image
	threshold	int	parameter of threshold
	length	int	parameter of length
	width	int	parameter of width
Output	*J*	matrix	the matrix storing the coordinate of the local maxima
	R	matrix	the matrix storing the information of matching results

The difference between this function and the function in Part II is that this function returns a matrix storing the information of each matching results.

Table 3.3 Description of DrawRectangle_DifferentSizes Function

Name	DrawRectangle_DifferentSizes

Description	Draw rectangles on the original image based on input matching results		
Input	Name	Type	Description
	I	matrix	the matrix storing the original image
	R	matrix	the matrix storing the matching results

The difference between this function and the function in Part II is that this function draws rectangles on the original image based on the input matching results matrix. In this matrix, for each matching result, coordinate of matching pixel and size of rectangle are saved.

Table 3.4 Description of GenerateCircleTemplate Function

Name	GenerateCircleTemplate		
Description	Generate a circle template with given radius		
Input	Name	Type	Description
	radius	int	the radius of the circle
Output	*T*	matrix	the matrix storing the template image

3.2 Generation of the template image

The template image is generated by drawing a circle of given radius. The results are stored in a matrix. The size of the template is 2**radius* + 1 X 2**radius* + 1. The image of a template image generated is shown in Fig. 3.1.

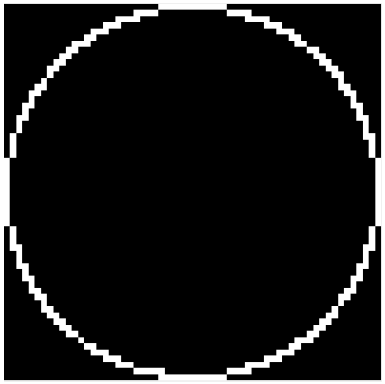


Fig 3.1 Template image with radius = 30 (pixels)

3.3 Matching over objects of different sizes

The basic idea is to generate template of different sizes and use Hausdorff method to match with each template. A matrix is used to store matching results. After a matching is completed, the current matching results will be compared with previous results. If a matching is detected in an area where a matching was already detected. Then the matching value of those two matching results will be compared. Larger one will be preserved and the smaller one will be dropped. If a matching is detected in a new area, then this matching result will be saved directly. Several steps are used to match over objects with different sizes.

Step 1: Generate the template image with given radius.

Step 2: Use Hausdorff method to match the input image with the generated template. In order to avoid too much repetition, the distance and coefficient is rather strict. In each iteration of matching, only several matching results will be detected.

Step 3: Compare the current matching results with previous saved matching results. Update the results matrix.

Step 4: Draw the rectangles to display each matching result.

3.4 Result of Matching

Test code:

```
I = imread('coins.png');                %read the input image and store it in matrix I
CoinMatching(I);                        %match input image with templates of different sizes
```

The matching results is shown in Fig. 3.2.

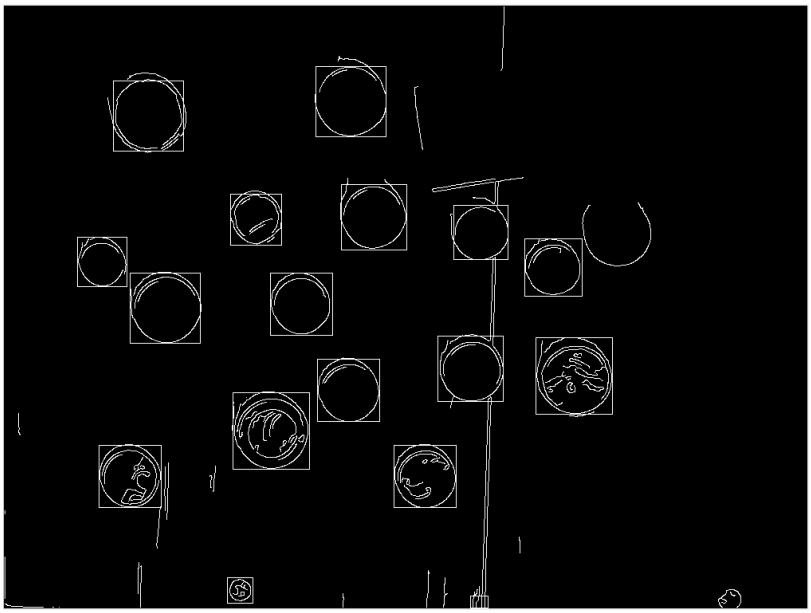


Fig 3.2 Result of coin matching

In this test, all coins and the nail in the bottom of the image can be detected. Actually, all closed circles were detected. The parameter of distance is 3 and coefficient is 0.9999, which are rather restrict. The radius of circle template is from 10 pixels to 50 pixels. If the radius range is changed, results will be different.

Actually, the image of cells can also be used to test the matching method. Almost all closed circles can be detected. The radius of circle template is from 10 pixels to 30 pixels. The result is shown in Fig. 3.3.

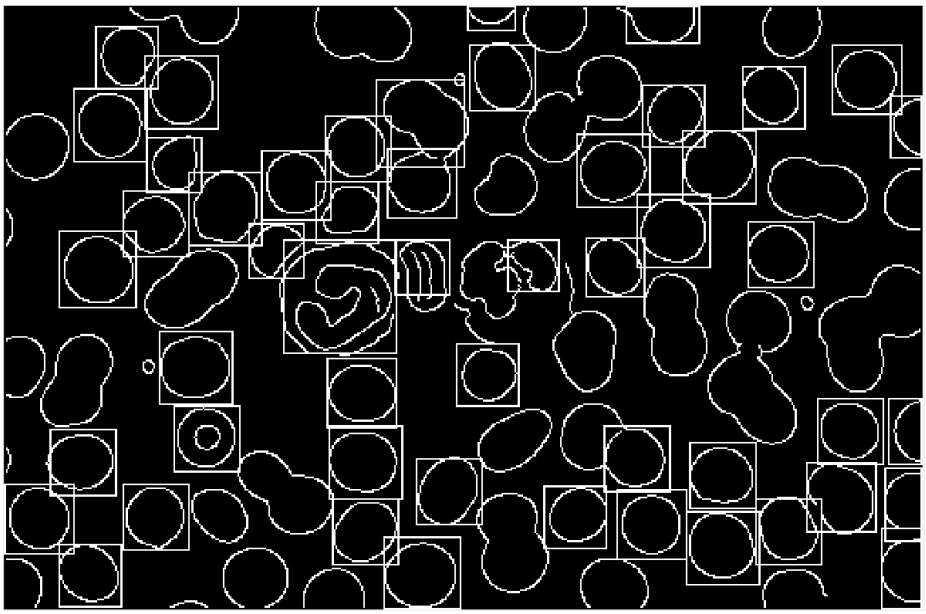


Fig 3.3 Result of coin matching used on “cells.png”