1. Largest inscribed circle

1 Introduction

The function LargestCircle will find the largest inscribed circle in any arbitrary image. This means some obstacles may be inside the image as shown in Fig. 10.

2 Instruction

LCout=LargestCircle(image,varargin)
Function to find the largest inscribed circle in an arbitrary image with multiple holes (black pixels).

2.1 INPUT:

image: Image, RGB, grey or BW, by preference BW.
E.g.: image=imread('C:\MyImage.tif');
Graphic: Optional, default: 1 (Plot graphic), 0: no graphic

2.2 OUTPUT

LCout:

- 1st value: Area of the largest circle in px
- 2nd value: radius in px
- 3rd and 4th value: x,y of the circle center. x: from left, y: from top of image

If area=0, black image (no circle found).

2.3 EXAMPLES:

Run and plot graphic: LCout=LargestCircle(myImage) ;
Run and do not plot graphic: LCout=LargestCircle(myImage,0)

2.4 REMARK:

Any hole (black pixel), even only one pixel wide, will not be inside the largest circle

2.5 Limits

None.

3 Method

3.1 Coordinate system

For image processing Matlab uses a coordinate system as shown in **Fig. 1**:

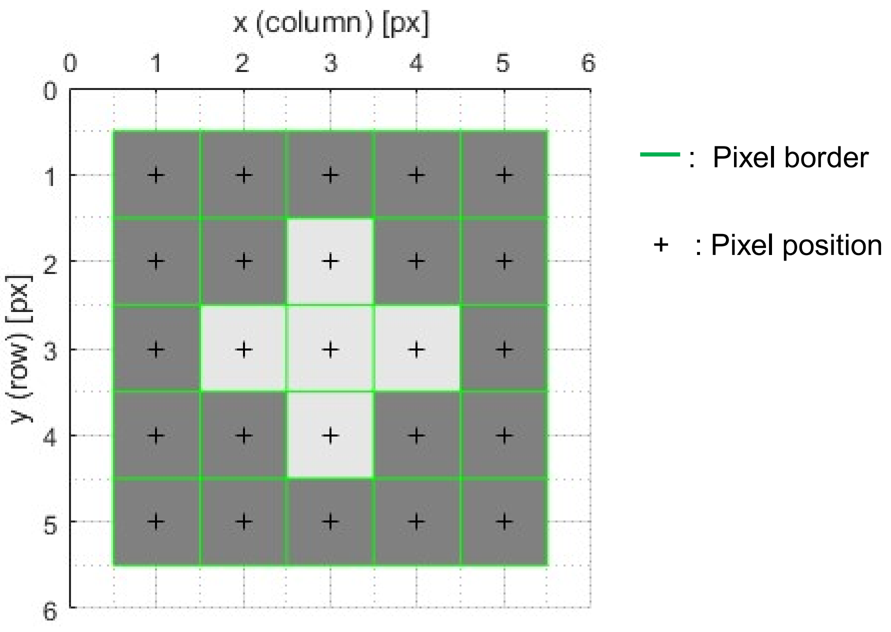


Fig. 1: Coordinate system

The origin of the coordinate system is outside the image at the top left corner.
The first pixel of the image is at position x,y=1,1.

3.2 Regular process

The regular process detects circles in areas that have at least one 4-connected element, see Fig. 2.

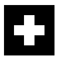


Fig. 2: 4-connected element

3.2.1 Boundary

After adding a black frame around the image (Fig. 3 and Fig. 4), the next step is to determine the white pixels that are next to one side of a black pixel, **Fig. 5**.

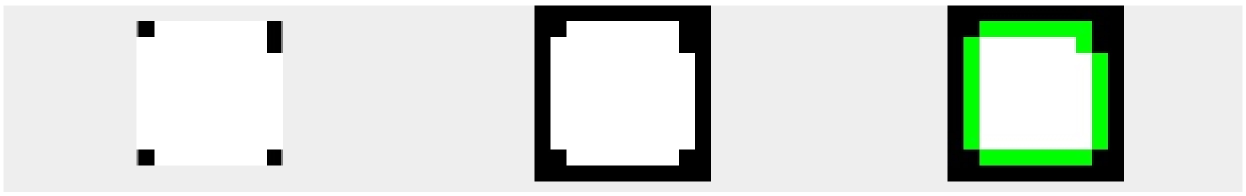


Fig. 3 Original image 9x9 px, (no frame)

Fig. 4 Image with added frame, 76 white px

Fig. 5 Boundary (green)

3.2.2 Main process

For all white pixels we calculate the distances of all boundary pixels to the current white pixel position.
In Fig. 6 we see the first 4 distances. The shortest distance determines the largest circle for this white pixel position.
At the white pixel with the position x,y=6,6 we get in the case of Fig. 7 the largest 'shortest distance'. The circle radius is here 4 px. The circle does not hit the black Border. In some cases, e.g. an image like Fig. 2, the circle goes through some black corners.

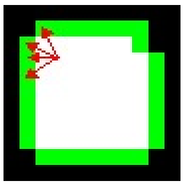


Fig. 6: Distances to one white pixel

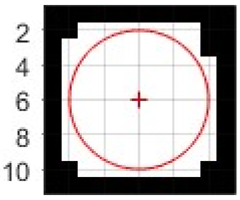


Fig. 7: Circle for integer measures

Therefore, some refinement is necessary.

3.2.3 Refinement

In order to get a circle that hits the border of the white area we add border positions as shown in Fig. 8 (red +) and recalculate circles with subpixel positions around the integer center of the so far detect largest circle and get the refined final circle, Fig. 9.

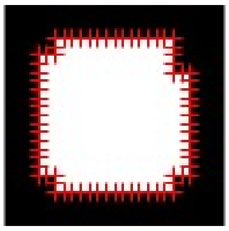


Fig. 8: New border positions

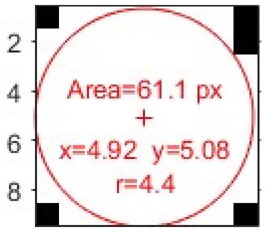


Fig. 9: Original image, final result

The previous steps, direct evaluation, are fast for small images.

3.2.4 Erosion

If we have large images, as in Fig. 10, the process to find all possible circles would take a lot of processing time. Therefore, the range for the possible circle centers is reduced by erosion of the image. Since the erosion itself is also costly, the image is downsized before erosion and then upsized to the original.
The erosion down to one pixel will not give us the center of the largest circle. The erosion is stopped at a certain percentage of white pixels, Fig. 11.

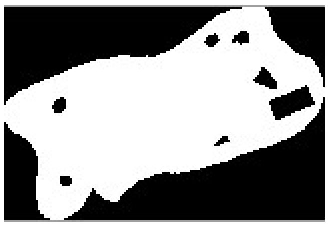


Fig. 10: Original big image, 215864 white pixels

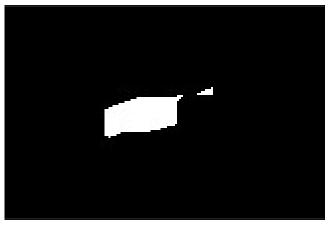


Fig. 11: Eroded image, 14141 white pixels

In the case of Fig. 10: Original big image, 215864 white pixels, the number of circles to evaluate is reduced from 215864 to 14141, Fig. 11. The white pixels of the eroded image, Fig. 11, determine the search range for the circle centers.
The circle search is then continued with 3.2.2 Main process and 3.2.3 Refinement.
The final result is:

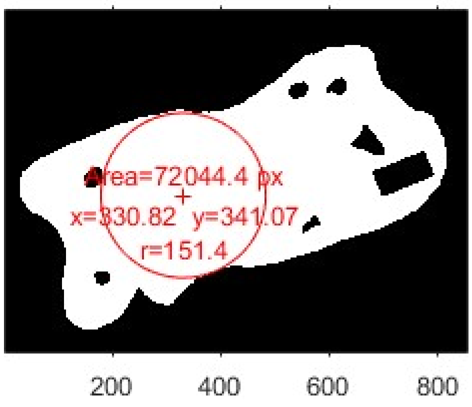


Fig. 12: Final result for the image of Fig. 10

3.3 Small circles

If only circles with radii of one are detected, many of them may be detected. One of the wrong detected circles is shown as example in Fig. 13.
Therefore, the search algorithm is refined to 0.5 px steps.

The process is:

- Enlarge image by two. This simulates 0.5 px steps.
- Erode image by 1 px for the search range.
- Do the main process on enlarged image, Fig. 14.
- Divide the results for radius and x,y center positions by two.
- Do the refinement process with the original image and get the final result, Fig. 15.

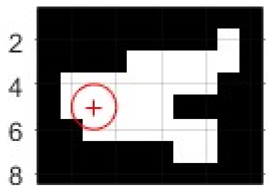


Fig. 13: Detected circle with r=1

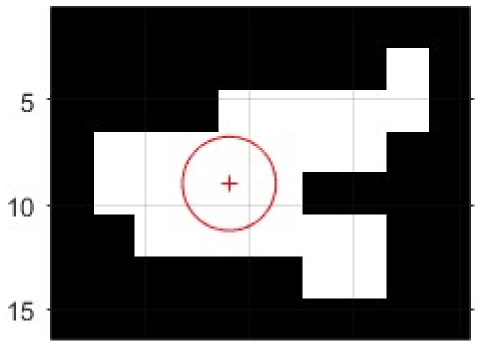


Fig. 14: Double enlarged image with detected largest circle

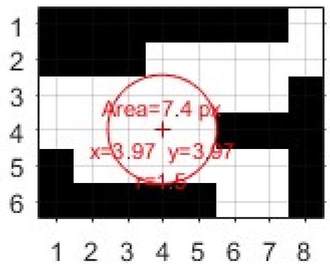


Fig. 15: Final result

3.4 Tiny circles

It may happen that on the first run no circle is detected, since the boundary is identical with the white pixels. Therefore the shortest distance is always zero.
Then a tiny circle must be present.
At least one of following shapes is inside the image:

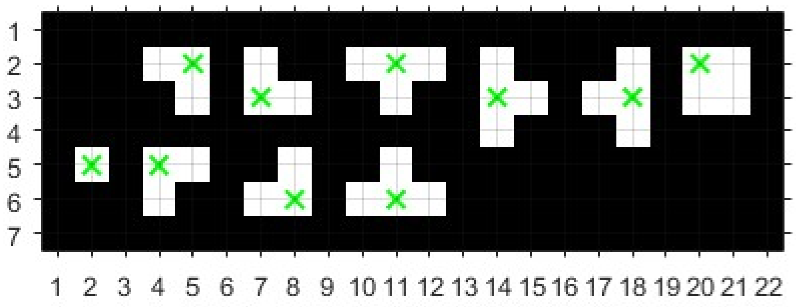


Fig. 16: Tiny shapes

The green crosses (x) indicate the reference positions.

All white pixels are tested for all shapes. E.g. for the corner shape, top left of Fig. 16, this corner type is found, if a white pixel is on the left and the bottom side of the reference position. The circle values are retrieved directly as presented in Table 1.

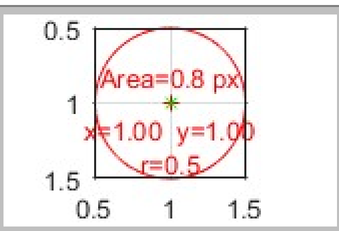
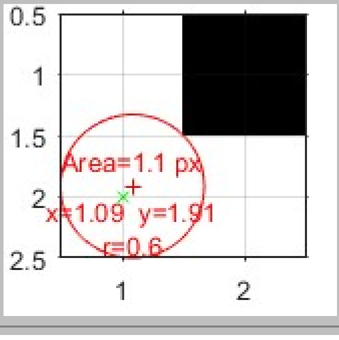
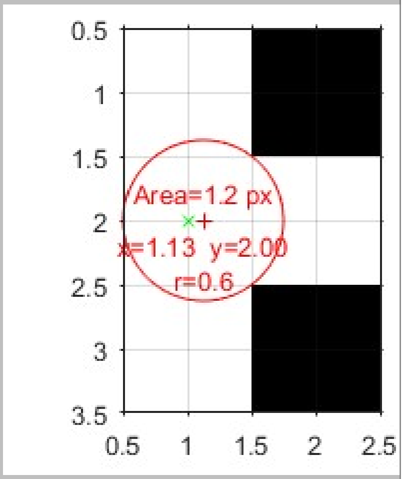
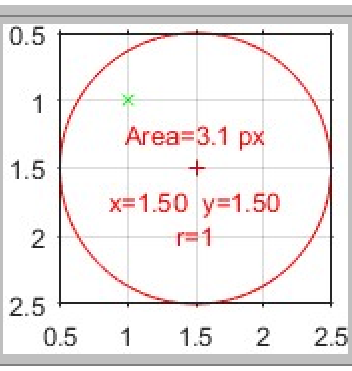
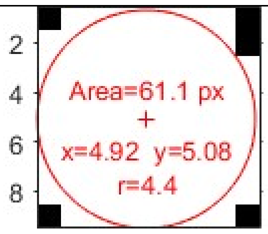
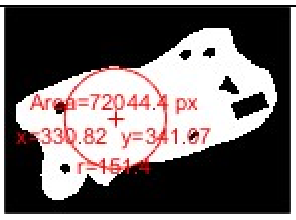
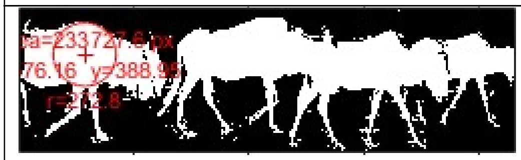
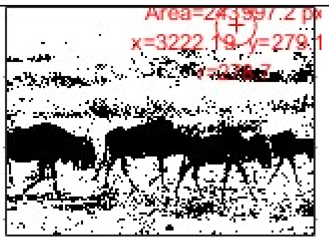
Tiny shape	radius	x-position	y-position
		of center relative to reference pixel (x)	
	r = 0.5	x = 0	y = 0
	$r = \frac{\sqrt{2}}{1+\sqrt{2}}$ r = 0.586…	x = r - 0.5	y = 0.5 – r
	r = 0.625	x = 0.125	y = 0
	r = 1	x = 0.5	y = 0.5

Table 1: Different tiny shapes and their circle parameters.

4 Processing time

The processing time is dependent on the image size, the content of the image and of course on the speed of your processor. Below are some examples for the running time on my PC.

Image	Image size	Process time
	9 x 9 px	20 ms
	629 x 856 px	0.6 s
	1221 x 4320 px	2.5 s
	3240 x 4320 px	139 s

5 Remarks

If you zoom into the plots, the pixels are not displayed correctly, at least on my PC. The size of the pixels may 10^{-7} too big or too small. The circle is displayed accurately.