IM520/MC505 Computer Vision
Term Report

Błażej Moska

March 22, 2020

Assignment 1

Circle detection in binary dot images

1.1 Introduction

The goal of this exercises was to find a points which could form a circle in a black-white image, consisting of random-noise. While the definition of the problem might not sound serious, research for possible solutions carried out.[2] proposes Harmony Search, an optimization method inspired by music.In [1] authors propose randomized algorithm, incorporating also edge detection.

1.2 Approach

To describe approach which was used to solve the problem, firstly some notation should be stated. A point in a 2 dimensional space is defined as follows:

$$p_i = (u_i, k_i)$$

where ıis an index of i-th point Equation of a circle is defined as follows:

$$(x - x_i)^2 + (y - y_i)^2 = r^2$$

where ıis an index of i-th point

The algorithm is based on collecting all of the points (that is to say, checking which pixel of an image is black, when we operate on white background or white, when the colors are reversed). In the next step random sample of three points is performed. Having three points selected, it is possible to obtain circle equation. Basing on [5] and high school mathematics materials available in the internet, such as [3], following equations are obtained:

$$x_i = \frac{(x_1^2 + y_1^2)(y_2 - y_3) + (x_2^2 + y_2^2)(y_3 - y_1) + (x_3^2 + y_3^2)(y_1 - y_2)}{2(x_1(y_2 - y_3) - y_1(x_2 - x_3) + x_2y_3 - x_3y_2)}$$

$$y_i = \frac{(x_1^2 + y_1^2)(x_3 - x_2) + (x_2^2 + y_2^2)(x_1 - x_3) + (x_3^2 + y_3^2)(x_2 - x_1)}{2(x_1(y_2 - y_3) - y_1(x_2 - x_3) + x_2y_3 - x_3y_2)}$$

$$r = \sqrt{(x - x_i)^2 + (y - y_i)^2)}$$

In every iteration of the algorithm basing on sampled points a circle equation is estimated. In further step it is verified how many points lay near formed circle. Some treshold must be put to measure closeness to the obtained circle. For this purpose following equation was used:

$$|(x - x_i)^2 + (y - y_i)^2 - r^2| \leq 0.5$$

Formula above describes how far the point could lay from the obtained circle, regardless of direction (inside or outside the circle) to be considered.

Algorithm 1.1: Algorithm pseudocode

Result: Best parameters for circle
Set following variables:

$$max\_number\_of\_points\_near\_circle = 0$$

$$best\_radius = 0$$

$$best\_x = 0$$

$$best\_y = 0$$

for $i \leftarrow 0$ **to** *number_of_iterations* **do**
  Randomly sample three points

$$number\_of\_points\_close\_to\_circle = 0$$

  if *if not collinear* **then**
    Calculate circle equation
    foreach *point in points* **do**
      if *meet criterion* **then**

$$number\_of\_points\_close\_to\_circle++$$

    end
    if *number_of_points_close_to_circle>max_number_of_points_near_circle* **then**
      max_number_of_points_near_circle=number_of_points_close_to_circle
      set best_x,best_y,best_radius
end

1.3 Results

Result of this assigment is an circle equation which satisfies the condition mentioned in previous section.

Analysis of the above figure leads to conclusion that the obtained circle is exactly as expected. After receiving best parameters it was drawn using function embedded in ImageJ drawOval , with these parameters.

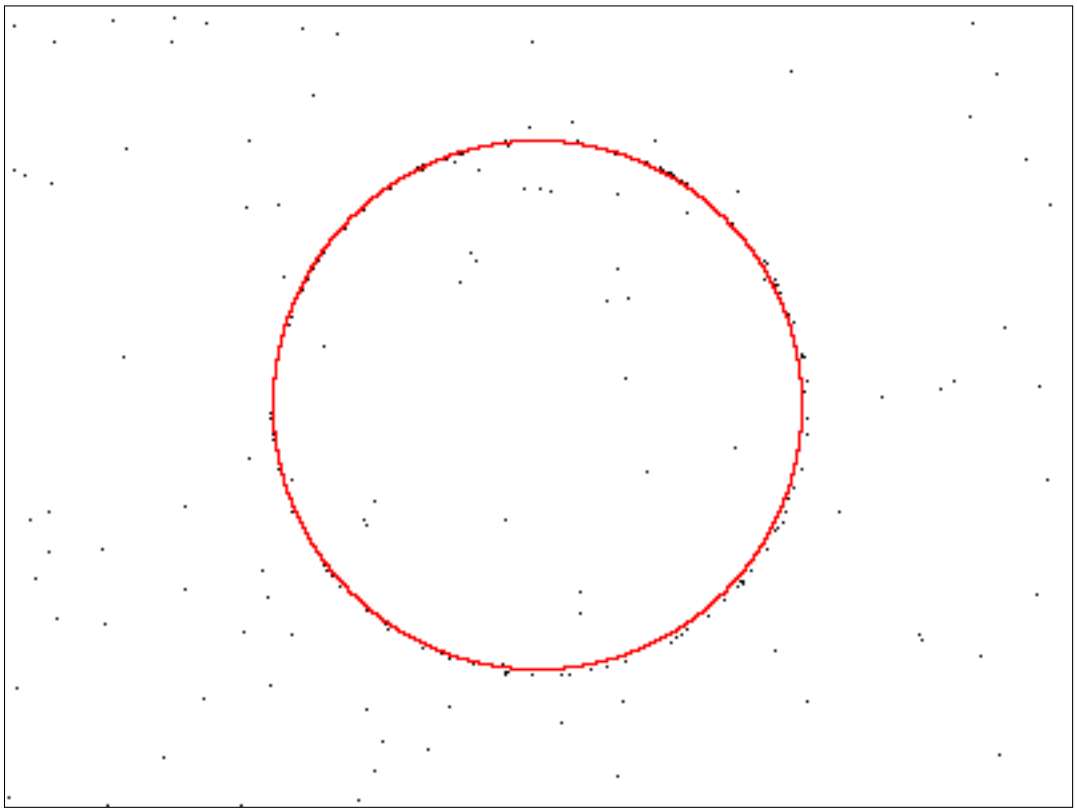

Figure 1.1: Obtained circle

1.4 Research question A

Given specified amount of points being total number of points, m and points which belongs to circle n, probability of drawing a 3 circle points in a random draw could be described as follows:

$$p = \frac{\binom{n}{3}}{\binom{m}{3}}$$

1.5 Research question B

The above algorithm could be somehow extended in such way that not only the best one with largest number of points in neighbourhood could be stored but a couple of them. If these circles are different then their parameters, that is center and radius should be different. The first circle with biggest number of points close to it and also with significantly different parameters can be potentially other circle.

Assignment 2

Affine Point Cloud Matching

2.1 Matching point sets by brute force or RANSAC

If your computer could perform 1 million such tests per second, how long would it take to examine all possible 3-point matches, if m = 10, 50, 1000

As can be seen on the figure above, number of possible combinations grows expotentially, since each point is matched with each another. When m=10 the experiment would last 1000 seconds, not even comparing to m=50 or m=100. This explains why RANSAC approach is almost completely useless in such case.

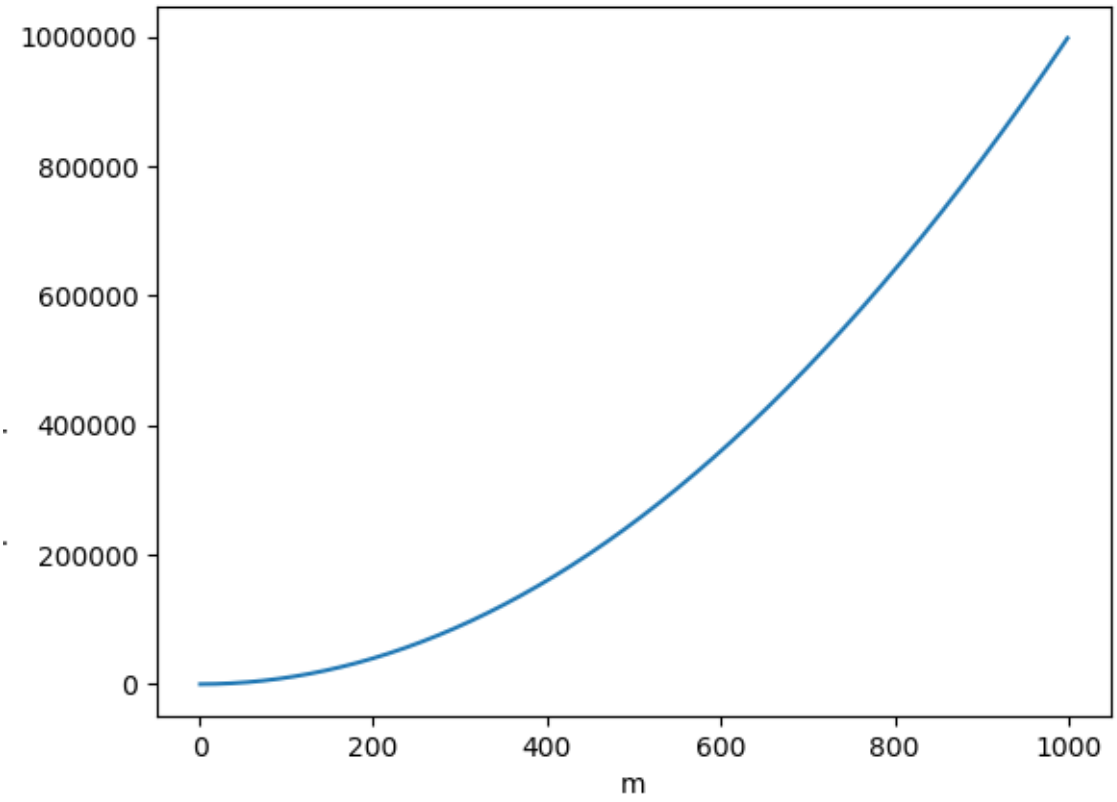

Figure 2.1: Obtained circle

2.2 Implement/test the affine transformation

Transformation indeed performed well, almost perfectly mapping set X to set X'. Almost, because there is a place for error because the results are not perfect pixel positions.

Affine transformation is:

$$\begin{pmatrix} 0.013 & 1.088 & 18.688 \\ -1.000 & -0.050 & 127.500 \end{pmatrix} \tag{2.1}$$

Using above matrix on X a X' is obtained. Results are presented in the ***Fig 2.3***:

Mapping error, i.e the difference between points mapped by affine transformation and real points is defined using following equation:

$$\epsilon_{total} = \sum_{i=0}^{n} \|\boldsymbol{A} * \boldsymbol{x_i} - \boldsymbol{x_i}\|^2 \tag{2.2}$$

Calculated error was as follows:

$$\epsilon_{total} = 5.72 \tag{2.3}$$

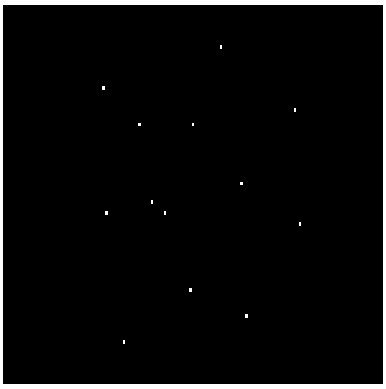

Figure 2.2: Transformed image

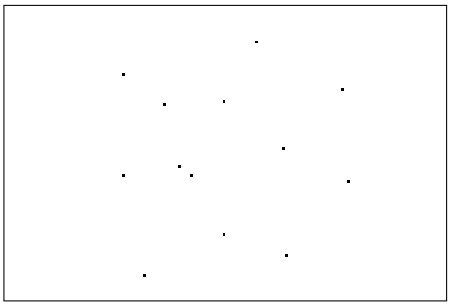

Figure 2.3: Real image

2.3 Structuring point sets by triangulation

Idea of triangulation

While using RANSAC is almost impossible because of computational reason, a way to overcome this problem is triangulation. Points from one set are structured using Delaunay algorithm. Given triangles in both set it is easier to perform calculations.

Algorithm 2.1: Algorithm pseudocode

Set following variables:

$$lowestError = Inf$$

$$bestAffine = emptyMatrix()$$

$$bestTransformedPoints = emptyMatrix()$$

Perform triangulation of X
Perform triangulation of X'

for $i \leftarrow 0$ **to** *number_of_triangles_in_X* **do**
  Take points from triangle from X dataset
  for $i \leftarrow 0$ **to** $number\_of\_triangles\_in\_X'$ **do**
    Take points from triangle from X' dataset
    foreach *permutation of points in X'* **do**
      calculate affine
      calculate transformedPoints
      map X to X
      calculate error
      if *error<lowestError* **then**

$$lowestError = error$$

$$bestAffine = affine$$

$$bestTransformedPoints = transformedPoints$$

Calculated matrix is as follows:

$$\begin{pmatrix} 0.019 & 1.096 & 17.827 \\ -0.98 & -0.047 & 126.541 \end{pmatrix} \tag{2.4}$$

Research Question

Affine invariancy, taken in simple words and according to [4], that applying affine transformation to triangulised data would be the same as triangulation of affine transformed data. This could be checked as performing triangulation on given X set and then applying affine transformation to compare whether it is the same as triangulation affine-transformed set X'.

Complexity

Complexity in this particular case is based on basic operation being solving system of equation. Having n points, n-2 triangles are formed in each set (X and X').We compare each triangle from set X with each of X', bearing in mind that permutations are considered (3 points in triangles, resulting in 6 triangles). Overall a square complexity is obtained $O(n^2)$

References

[1] Yazan Alomari, Siti Norul Huda Sheikh Abdullah, and Khairuddin Omar. "Randomized Circle Detection Performance Based on Image Difficulty Levels and Edge Filters". In: *FIRA RoboWorld Congress*. Springer. 2013, pp. 361–374 (cit. on p. 2).

[2] Jaco Fourie. "Robust circle detection using Harmony Search". *Journal of Optimization* 2017 (2017) (cit. on p. 2).

[3] Robert Eisele. *Create a circle out of three points*. URL: https://www.xarg.org/2018/02/create-a-circle-out-of-three-points/ (cit. on p. 2).

[4] Unknown. *what-does-invariant-to-affine-transformations-mea*. URL: https://math.stackexchange.com/questions/2634253/what-does-invariant-to-affine-transformations-mean (cit. on p. 8).

[5] Wikipedia contributors. *Ellipse— Wikipedia, The Free Encyclopedia*. [Online; accessed 22-July-2004]. URL: https://en.wikipedia.org/wiki/Ellipse#Circles (cit. on p. 2).