IM520/MC505 Computer Vision
Term Report

Peter A. Wiseguy

March 21, 2020

Abstract

This document is adapted from the `HgbTermReport` template and based on the `hgbreport` LaTeX class, which is part of the HagenbergThesis document package. See https://github.com/Digital-Media/HagenbergThesis for the most recent version and additional materials (tutorial, manual etc.). Use this *Abstract* to provide a short summary of the contents in the remaining parts of the document. Note that it may be easier to place the individual chapters ("assignments") in separate files and include them using `\include{..}`.

Contents

Guidelines for authoring lab reports

Cumulative lab report

The lab report is a **single, cumulative document** which should contain a concise and well-structured summary of the work you did in this course. Also, you are asked to demonstrate and discuss your "report in progress" at any time throughout the semester. If help or advice is needed, please ask in class or use the course's online forum.

Weekly and final submissions

You are asked to upload a snapshot of your worked-out assignments weekly (i.e., prior to the next lab unit). These submissions are not graded but randomly checked to verify your progress. The final (complete) documentation for all assignments must be turned in at the end of the semester, prior to the (oral) exam. Thus you can pace your work individually and turn back to previous assignments for improvements at any later point.

> **Note** that this **freedom** puts a lot of **responsibility** on yourself. Make sure that you start to write immediately, make steady progress and nothing important is left behind!

Document structure and content

This document should help you to get started with the report. It is strongly suggested to use the final format right away to avoid surprises at a later point. Also, you will discover that writing and documenting your findings can help you in developing good and understandable solutions from the very beginning. Here are a few hints for writing your reports:

- One **chapter** should be dedicated to each **assignment** (note that chapter names have been modified for this).
- Make notes and write down your concepts immediately, that is, **before** you start coding!
- Describe each given task in your own words (do not just replicate the assignment). Then describe your approach, explain the main difficulties, clearly outline your solution, finally provide illustrative and meaningful results.
- Try to go beyond the material you find elsewhere, use and extend formal (mathematical) descriptions in a creative way. Also, try to keep your notation simple and

consistent, which is not always easy to do. Look at good examples and consider this part of the learning process.

- Be careful and creative when it comes to designing meaningful tests and selecting examples. Do not make screenshots but save the relevant images with ImageJ (usually as PNGs).
- Always give appropriate references to literature, figures and other work you used!
- Get used to work with formal and concise descriptions (math, symbols, relations, algorithms, ...) and train yourself in "getting the notation right".
- Write in complete sentences and try to use a "professional" language.

The bad and the ugly

- **Don't just show program code!** Use prose with mathematical and algorithmic notation wherever appropriate (use the assignments and lecture notes for guidance). Insert actual code sparingly and only to show particularly interesting or critical parts of your implementation.
- **Do not explain details that are trivial** or elementary (such as Pythagoras' law, for example). Otherwise, make a reference to the *all* sources you used (including school books, blogs, WikiPedia etc.).
- **Do not just replicate** equations and figures from the lecture materials, but – as said above – describe the task in your own words. In particular, you will be **executed** (i.e., beheaded, drawn and quartered) if you ever copy/paste equations from the assignment or any other sources. Make sure you write these things yourself (that's what LaTeX is famous for)!

Assignment 1

Circle detection in binary dot images

In this assignment we want to detect a circle which is embedded in a noisy binary image. To accomplish this task the RANSAC algorithm is used.

1.1 Algorithm

At first the coordinates of all black pixels are collected and gathered in a list of points. Out of this list 3 points are randomly chosen and we make sure that no point is selected more than once. With the help of these 3 points it is possible to determine the attributes of a circle which passes through them as shown in Figure 1.1 following an article from Paul Borke[2].

We form one line through the point P_1 and P_2 and another line through P_2 and P_3.

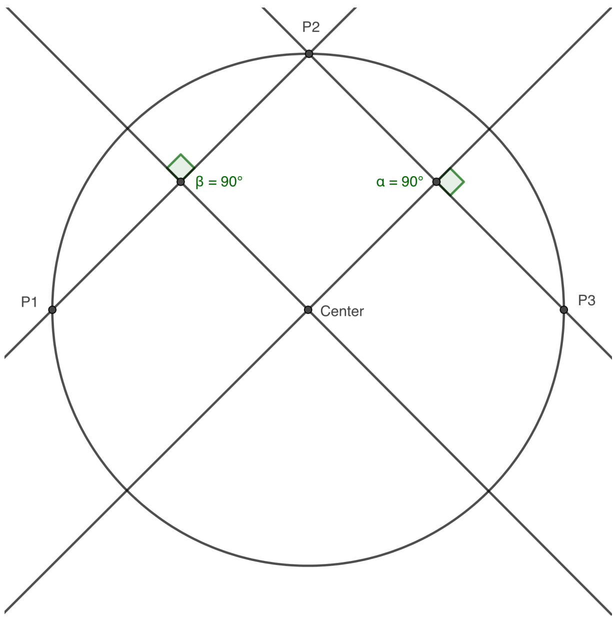

Figure 1.1: Determine the center of a circle with 3 points.

The equation for these two lines are:

$$y_a = m_a(x - x_1) + y_1$$
$$y_b = m_b(x - x_2) + y_2$$

m is the slope for each line. So we have:

$$m_a = \frac{y_2 - y_1}{x_2 - x_1} \qquad m_b = \frac{y_3 - y_2}{x_3 - x_2} \tag{1.1}$$

Now two lines perpendicular to the lines $y'_a(P_1P_2)$ and $y'_b(P_2P_3)$ going through the center between each point pair are created. The perpendicular of a line with slope m has slope of $-1/m$. This results in following equations:

$$y'_a = -\frac{1}{m_a}(x - \frac{x_1 + x_2}{2}) + \frac{y_1 + y_2}{2} \tag{1.2}$$
$$y'_b = -\frac{1}{m_b}(x - \frac{x_2 + x_3}{2}) + \frac{y_2 + y_3}{2} \tag{1.3}$$

The center of the circle is the intersection of these two perpendiculars.

$$x = \frac{m_a m_b(y_1 - y_3) + m_b(x_1 + x_2) - m_a(x_2 + x_3)}{2 \cdot (m_b - m_a)} \tag{1.4}$$

To calculate the y value of the center I substitute the x value into one of the two perpendiculars (1.2, 1.3). The circle radius is determined by calculating the distance between the center point and one of the 3 originally chosen points.

There are 3 situations where a circle can not be calculated with just 3 points:

- If all 3 points are collinear.
- If one point is selected more than once. This gets checked while selecting the points.
- If either line is vertical their slope would be infinite. To avoid that the order of the 3 points is rearranged if this is the case.

After the circle is calculated the number of points on the circle gets determined. The distance between each detected point of the image and the center point of the circle is calculated and its difference to circle radius. If this difference is smaller than a certain threshold, the point is on the circle and it is saved in the point list of the circle. The threshold can be defined when the plugin is started.

The circle is saved as currently detected circle and this whole process gets repeated for a certain number of times which can be defined at the start of the plugin. After each iteration the number of points on the calculated circle is compared to the number of points on the currently detected circle. If the count is higher the detected circle gets replaced by the calculated circle.

As soon as this process was repeated for the defined number of times. The current detected circle gets drawn on the image in blue as well as all the points on it in red.

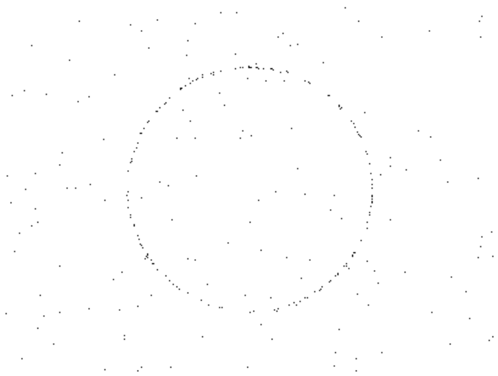

Figure 1.2: Generated noisy binary image.

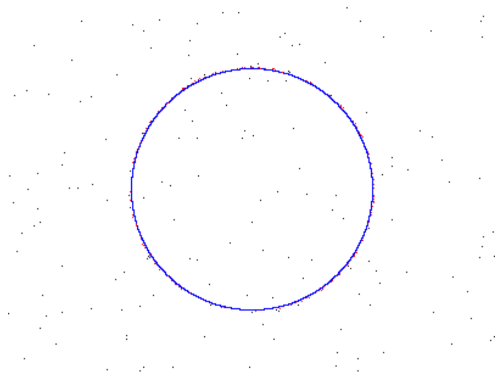

Figure 1.3: The Detected circle after 25 iterations and a threshold distance of 1 from the circle

1.2 Research Question A

- The probability of selecting one point which lies on the circle is: $\frac{m}{n}$. If we choose 3 random points in a row we also need to make sure that we do not select the same point twice. This results in following formular:

$$P = \frac{m}{n} \cdot \frac{m-1}{n-1} \cdot \frac{m-2}{n-2} \tag{1.5}$$

- To calculate the number of random draws n needed to select 3 circle points with a probability of 99% we use following formular: $W = 1 - (1 - p)^n$ [1]. To calculate p use the formular above (1.5) then solve the equation to get n.

$$0.99 \geq 1 - (1 - p)^n \tag{1.6}$$
$$0.01 \geq (1 - p)^n \tag{1.7}$$
$$n \geq \frac{\ln(0.01)}{\ln(1 - p)} \tag{1.8}$$

1.3 Research Question B

To detect multiple circles in one image a threshold for the number of points needs to be defined which are needed to be on a circle to be detected as a valid circle. If detected the circle is added to a list. To make sure that the same circle is not detected multiple times it has to be checked that there is no circle with the same center point and radius already in the list before adding it.

Assignment 2

Affine Point Cloud Matching

2.1 Matching point sets by brute force or RANSAC

2.2 Implement/test the affine transformation

2.3 Structuring point sets by triangulation

Summary

Finally, summarize what has been accomplished in this semester and what not. Point out topics that were instructive, confusing, too hard, too easy etc. Perhaps you even found problems that you would like to explore deeper (e.g., in a project).

References

[1] *Diskrete Wahrscheinlichkeit*. Mar. 2020. URL: https://www.mathe-online.at/materialien/Daniela.Eder/files/Diskrete_WK/Zusammenstellung.html (visited on 03/12/2020) (cit. on p. 7).

[2] *Equation of a Circle from 3 Points (2 dimensions)*. Mar. 2020. URL: http://paulbourke.net/geometry/circlesphere/ (visited on 03/12/2020) (cit. on p. 5).