IM520/MC505 Computer Vision
Term Report

Peter A. Wiseguy

March 12, 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

Detect a circle out that is embedded in a noisy binary image.

1.1 Algorithm

- Collect the coordinates of all black pixels
- Randomly choose 3 points and use RANSAC Algorithm to detect points
- Calculate parameters of the circle (center point(x,y), radius) out of the 3 points
- Count the points that are sufficiently close to the circle (certain distance parameter)
- Repeat the steps above and save the points of the best fitting circle (circle with highest point count)
- draw the circle onto the input image and output its parameters

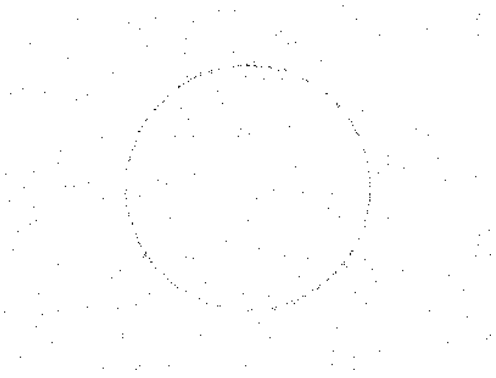

Figure 1.1: Generated picture

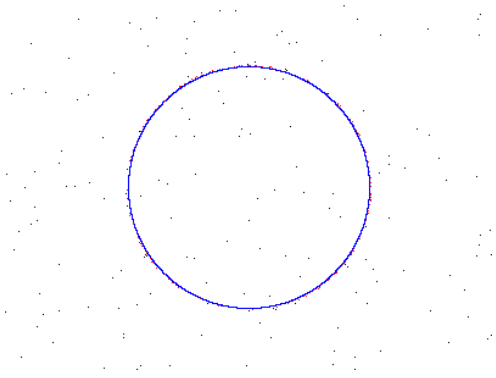

Figure 1.2: Result picture

1.2 Research Question A

- The probability of selecting 3 circle points in a random draw if we make sure that we don't draw the same point twice is:

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

- There are at least 7 random draws needed to detect 3 circle points.

$$W = 1 - (1-p)^n$$

1.3 Research Question B

To detect multiple circles in one image we need to define a threshold for the number of points 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 we need to check before adding it to the list that there is no circle with the same center point and radius already detected.

Assignment 2

Title of Assignment 2

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