ELEN-0016 – Computer Vision
Student projects 2019-2020

Prof. M. Van Droogenbroeck and Ph. Latour

Version 1.2

Contents

1 Introduction

The aim of the project is to design methods, to understand its components, and evaluate the quality of the results for an application.

The application consists to manage the image/video acquisition process and then to detect lines and elements with an elliptical shape.

Elliptical structures are pervasive in Computer Vision application. From one side, the image representation (projection) of any circular object of the real world is an ellipse; for instance the wheels of automotive vehicles or the traffic signs, the central circle of sport fields, some parts/sections of manufactured objects, the pupil/iris of eyes, etc. From another side, numerous objects of interest are often represented by oblong blobs (approximately elliptic); for instance the human face or body parts, cells or tumors in medical applications, galaxies in astronomical images, etc. Ellipses may also be considered as parts or building blocks of more complicated structures.

1.1 Organization of the working teams

Each team consists of 4 or 5 persons.

If possible, each team should be composed of students with different orientations (electronics – computer sciences – biomedical engineering). It is believed that such a team composition will enrich your experience and lead to the best results.

The teams must be constituted and the project will start on **Friday, the 4 October 2019**. Each team must send an e-mail (with all the team members in cc) to `mailto:philippe.latour@uliege.be` no later than **Friday, the 4 October 2019, at 17h00.**

1.2 Project subdivision

The project is about detecting lines and finding elliptical objects in images. For this kind of task (object matching), there is mainly two approaches; pixel (area) based or edge (feature) based. Also note that, when extracting ellipses from their contour (edges), a confusion often arises between straight lines and very flat arcs of ellipses.

The project will be subdivided in two parts;

Part 1 The first part of the project is devoted to the development of an application for image/video processing and the detection/extraction of straight lines only. Line detection is, in itself, an important building block of numerous applications like 3D scene analysis. But, it may be used prior to ellipse detection, in order to limit the confusion between lines and flat ellipse arcs. We expect you to use classical methods for this first task.

Team	Members	Registered
1	BUERES Y DOMINGUEZ Lisa	Yes
	DEBES Baptiste	Yes
	DEFRAIRE Stephan	Yes
	DIA Amadou Sall	Yes
	WEYDERS Pierre-François	Yes
2	L'HOEST Julien	Yes
	MIFTARI Bardhyl	Yes
	POLET Quentin	Yes
	ROEKENS Joachim	Yes
	STASSEN Théo	Yes
3	BOURDOUXHE Alexandre	Yes
	DESJARDINS Jérémie	Yes
	HOCKERS Pierre	Yes
	SERON Damien	Yes
	SIMAR Julien	Yes
4	MEURISSE Maxime	Yes
	ROZET François	Yes
	RUMFELS Océane	Yes
	VERMEYLEN Valentin	Yes
5	BERNARD Simon	Yes
	JANQUART Justin	Yes
	KLAPKA Ivan	Yes
	SCHOFFENIELS Adrien	Yes
6	KOUTCHEME Charles	Yes
	LE Ba	Yes
	LIBERT Robin	Yes
	ZIANS Dominik	Yes
7	BLISTEIN François	Yes
	LHOEST Alexandre	No
	ROUSSEAU Antoine	Yes
	VANDEGHEN Renaud	Yes
8	EL OSROUTI Mohamed	Yes
	HORBACH Amadis	Yes
	LEJEUNE Gary	Yes
	PAQUAY Joachim	Yes
	SPITS Martin	Yes
9	BOLLAND Adrien	Yes
	DELAUNOY Arnaud	Yes
	MINNE Adrien	Yes
-	CALIXTE Maxence	Yes

Table 1: Table of the teams

Part 2 The second part aims at the extraction of ellipses and the discussion of the results and performances you obtain. We expect you to develop methods based on machine learning for this second task.

1.3 Development platform

For each part, you will have to write, put into place and evaluate programs written in python 3 (possibly based on `OpenCV` and/or other libraries) running on a Linux platform. Pay attention that you will have to report your work with a `jupyter` notebook (see section 1.4.1) and that python 3.7 is not (yet) fully compatible with `jupyter` notebook. Therefore, we encourage you to used python 3.6.

1.4 Evaluation

Each part will give rise to a report and a presentation. The schedule for the reports and presentations is given in the following chapters.

1.4.1 Reporting

The report for each part must contain:

- A `jupyter` notebook (NO PDF!) with:
  - A short presentation of the part.
  - The contribution of each student of the team with respect to the 4 tasks. Several students may work on the same task and a student may work on several tasks. But we expect the work to be reasonably and fairly distributed.
  - For each tasks, you may present, if applicable;
    - A short description of the implemented algorithms, their advantages and drawbacks.
    - A short note on the implementation and of the validation test.
    - A few results of the execution of your code on representative examples and the corresponding discussion.
    - References to any information/inspiration sources you used (scientific papers, web `sites`, available libraries/modules, ...).
  - Please note that it is mandatory, for the first part (resp. second part), to limit the size of your notebook to **10 (resp. 15) pages maximum** (including title page, figures, images, tables, graphics, code execution results, etc.). Pages beyond 10 (resp. 15) pages will be discarded. You should use the print preview function of your browser to count the number of pages of your notebook.

- The code of the applications/modules produced by the four/five students in the team, **with a clear description** of how to use the code (install, run and test). This code will be used by the `jupyter` notebook.
- Optionally, the images/annotations acquired/used for this part.

1.4.2 Presentation and demo

You are strongly encouraged to showcase your applications/modules during the presentation.

Your presentation time is strictly limited to 20 minutes to present the project, including the time for a visual demonstration of your application.

All the members in the team have to present a part of the project.

1.5 Schedule

Date	Time	Milestones	Room
Friday, 20/09/2019	8h30-12h30	Theory	2.93
Friday, 27/09/2019		Day-off	
Friday, 04/10/2019	8h30-12h30	Theory + Student project presentation	2.93
Friday, 11/10/2019	8h30-10h30	Theory + Exercises	2.93
Friday, 18/10/2019	8h30-12h30	Theory + Exercises	2.93
Friday, 25/10/2019	8h30-12h30	Theory + Project	2.93
Friday, 01/11/2019		Day-off	
Wednesday, 6/11/2019	17h00	Report of the first part of the project	R87a
Friday, 08/11/2019	8h30-12h30	Presentation of the first part of the project	2.93
Friday, 15/11/2019	8h30-12h30	Theory+ Project	2.93
Friday, 22/11/2019	8h30-12h30	Theory + Image Database & Annotation	2.93
Friday, 29/11/2019	8h30-12h30	Theory + Exercises	2.93
Wednesday, 4/12/2019	17h00	Report of the second part of the project	R87a
Friday, 06/12/2019	8h30-12h30	Presentation of the second part of the project	2.93
Friday, 13/12/2019	8h30-12h30	Theory	2.93
Friday, 20/12/2019	8h30-12h30	Theory	2.93

2 Part 1: Main modules development and line extraction

2.1 Description

This part aims to develop the main modules needed to manage/process images/videos from disks and to detect lines in images. This part is further subdivided into 4 tasks described hereafter.

2.1.1 Preliminary remarks

For the four tasks in this part;

- You have to write and put into place a program(s) written in python 3, possibly based on `OpenCV` and/or other libraries, running on a Linux platform. As explain in section 1.3, for compatibility reasons with `jupyter` notebooks, you should use python 3.6 (and NOT python 3.7).
- You are requested to provide *a list of all the parameters of the method(s)* that you use/develop and, next, to discuss their impact/influence on the results.
- You are allowed to reuse any algorithms available in `OpenCV or other` libraries. We expect you to understand not only the theory behind the algorithms but also their advantages, drawbacks and their internals.

2.1.2 Task 1.1: Image/video processing application

- For this part, we will provide sequences of test images (1280 by 960, '`png`' format), but, in addition, you are free to acquire/use your own image sequences to test and showcase your application.
- You will work with gray-scale and/or color images. You should preferably use gray-scale images for performance reasons. Processing color images is more complex than to handle gray-scale images, also for the task of line extraction. Nevertheless, if you think that using color images is worth the extra cost, you are allowed to do so and to discuss the advantages and drawbacks of the color images compared to gray-scale images.This application (often called video processing loop) will be able to read and process from the disk/memory any video or image sequences.
- If needed, you may also reduce the image resolution before passing it to the processing module.
- In this task, you may want to *display* the current image and the results of the processing.
- Processing the image is done, not in Task 1.1, but in other tasks.

2.1.3 Task 1.2: Edge points extraction

- The program will be able to extract local edge (or feature) points from the images.
- You are free to choose (and/or combine) the method(s) (gradient, Canny, moments, ...).
- You should pay attention to the image filtering process (scale choice) which is often part of most of the methods used for edge points extraction.

2.1.4 Task 1.3: Line (segment) detection

- The program will be able to detect lines (and/or line segments) from the local edge points list extracted previously. There are thus two levels in this task;
  1. detecting the lines supporting the segments and
  2. detecting the end-points of the segments on the lines.
- You are free to choose (and/or combine) the classical method(s) (Hough, RANSAC, contour following, ...). Depending on the method you will choose you may obtain first the support lines (Hough, RANSAC) or the segments (contour following). When you consider this task as a prerequisite for the next part (discarding edge points belonging to a line for the ellipse detection process), the most important aspect is to obtain the support line.

2.1.5 Task 1.4: Edge points classification

- The program will be able to classify local edge points previously extracted as belonging to a line (segment) or not.
- You are free to choose (and/or combine) the method(s).

2.2 Evaluation

2.2.1 Reporting

The report is due for **Wednesday, the 6th of November no later than 17h00**, by mail sent to philippe.latour@uliege.be, or on a USB key in the R87a room.
As explained in section 1.4.1, the report must contain:

- A `jupyter` notebook (NO PDF!) of maximum 10 pages
- The code of the applications/modules produced by the four/five students in the team, **with a clear description** of how to use the code (install, run and test).

- Optionally, the images you already acquire during the task 1.1 to test your algorithm.

2.2.2 Presentation and demo

The presentation will take place on **Friday, the 8th of November starting at 8h30** in the 2.93 room.

You are strongly encouraged to showcase your applications/modules during the presentation and all the members in the team have to present a part of the project.

Your presentation time is strictly limited to 20 minutes to present the project, including the time for a visual demonstration of your application.

The schedule for the presentation is

Team	Time
1	8h30-8h50
2	9h00-9h20
3	9h30-9h50
4	10h00-10h20
5	10h40-11h00
6	11h10-11h30
7	11h40-12h00
8	12h10-12h30
9	12h40-13h00

3 Part 2: Ellipse detection and performance assessment

3.1 Description

Description not available yet.