ELEN-0016 – Computer Vision
Student projects 2019-2020

Prof. M. Van Droogenbroeck and Ph. Latour

Version 1.4

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	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	Yes
	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
10	BUERES Y DOMINGUEZ Lisa	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, ...).
  - Your conclusions, comments and ideas for further enhancements of the application.
  - 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). The results of the execution of the

code cells are not included in the page count. So you are allowed to clear the results before sending the notebook. 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.
  - Your code should not be entirely in the notebook. Instead, we expect you to import in the notebook the code you developed and that you want to demonstrate in the 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 should be imported and used in 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	Project report (first part)	R87a
Friday, 08/11/2019	8h30-12h30	Project presentation (first part)	2.93
Friday, 15/11/2019	8h30-12h30	Theory+ Project	2.93
Thursday, 21/11/2019	23h59	Image Annotation	Cytomine
Friday, 22/11/2019	8h30-12h30	Theory	2.93
Friday, 29/11/2019	8h30-12h30	Theory + Exercises	2.93
Friday, 06/12/2019	8h30-12h30	Theory	2.93
Thursday, 12/12/2019	23h59	Project report (second part)	email
Friday, 13/12/2019	8h30-12h30	Theory	2.93
Friday, 20/12/2019	8h30-12h30	Project presentation (second part)	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 ('`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 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).

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 Image Database

The image database contains 500 images of five different types.
It is perfectly allowed (and somehow suggested) to adapt your algorithm (different parameters or methods, different preprocessing, different post-processing) depending on the type of images you have to process.
Indeed, we expect you to provide a specially adapted version of your algorithm for (at least) two of the image types and to simply analyze the results of your methods on the other image types (and to quickly discuss what could be done to obtain better results).

2.2.1 Building

There are 95 images representing, most of the time, buildings with several floors and windows. Two sides of the buildings are visible on the images. There are 19 different buildings and 5 views (images) per buildings.

The objective of line detection on these images is to obtain the main directions (two or three) as a preprocessing for 3D scene reconstruction. A second step could be, when possible, to obtain a maximum of the segments belonging to the contours of the windows and the building.

2.2.2 PCB

There are 101 images representing from 3 to 5 views of 23 PCB.

The objective on these images is first to obtain the borders of the PCB and secondly to obtain a maximum of the segments of the contours of the main elements on the PCB (sockets or components).

2.2.3 Road

There are 113 images of roads with traffic signs, vehicles, pedestrians and some buildings.

The main objective is to obtain the lines limiting the road and the traffic lanes when visible. The dotted lines may be extracted as continuous lines or lists of line segments (it is your choice).

A more challenging objective, would be to extract the line segments delimiting the traffic signs.

2.2.4 Soccer

There are 92 images of a soccer field with the players and the public.

The objective is to obtain the field lines and possibly the goal lines.

2.2.5 Sudoku

There are 99 images of sudoku grids (33 different grids and 3 views per grid).

The objective is clearly to obtain the separating line segments of the grids (10 horizontal and 10 vertical line segments).

2.3 Evaluation

2.3.1 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 (you should do a print preview of the notebook in your browser and look at the page number of the produced document).
- 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.

2.3.2 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 matching and performance assessment

3.1 Description

This part aims to develop the modules needed to detect/match ellipses in images and to assess the performances of some of the modules you developed. This part is further subdivided into 4 tasks described hereafter.

3.1.1 Preliminary remarks

For the four tasks in this part;

- You have to write and put into place program(s) written in python 3, possibly based on `OpenCV`, `Keras` 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, Keras or other` libraries or any existing already trained network. We expect you to understand not only the theory behind the algorithms but also their advantages, drawbacks and their internals.

Generally speaking for a performance assessment task;

- You should study, clearly design and describe how you are going to evaluate the performances (methodology, typology, kind of evaluation task, criteria/scores, reference data, ...).
- It can be a qualitative assessment and/or a quantitative assessment.
- If you come up with a binary classification problem, pay attention to correctly define and use the four quantities (TP, FP, FN, TN) of the confusion matrix.

3.1.2 Image database

In the context of this part of the project, we will provide two specific image types:

1. Soccer field images (color); where you will have to extract the white circles of the field marking (one central circle and two side half-circles). We will provide a database of at least 1500 real images (1920x1080) of the soccer field lines and circles.

2. Eye infra-red images (gray-scale); where you will have to find the pupil. We will provide a database of at least 3 000 real small images (320x240) of the eye in close-up.

It is obviously allowed to reduce the size of these images if needed.

3.1.3 Task 2.1: Performance assessment of line segment detection

This task is aimed at assessing the performances of the line segment detection module.

- It is a quantitative assessment, where you will compare the results of your line segment detection modules (developed during the first part of the project) with the ground truth on some annotated images.
- You are free to choose/design the comparison metrics which seem the most appropriate in this case.
- The image annotation process (ground truth) is described in section 3.1.5.
- Practically, you will write a program to implement your metrics and perform the comparisons on all the images annotated by all the students.
- In your report, you will describe the metrics used and you will present the results obtained on these annotated images.

3.1.4 Task 2.2: Ellipse matching

The main objective of this task is to detect and locate elliptical structures in images. There are a lot of situations in which this could be useful as a preprocessing or an almost final step of processing.

Conceptually, *matching ellipses* means that you have first to decide if (and how many) ellipses are present in the image (detection) and then to provide more or less accurate values for their parameters (location, orientation, size). Most of the time, both operations are achieved together. Indeed, a good way to detect an ellipse in an image is to provide its parameters, but keep in mind that these are two different problems.

Pay attention that, *fitting an ellipse* is another problem in which we assume that we already know a list of points belonging (almost) for sure to one ellipse (almost no noise or outliers in the point list). Then, ellipse fitting consists solely in computing the best and accurate value for the parameters of one ellipse. In this project, you have most of the time a lot of outliers or noise in the images and, in addition, you are not sure if there are ellipses, nor their numbers, in the image.

Machine learning. We expect you to use machine learning approaches to solve this problem. Nevertheless, in addition to the neural network modules, it is perfectly allowed to also use classical image processing algorithms. When designing your system, you have to pay attention to the architecture of the network and to the feasibility of the learning process:

- Should you design a completely home-made network and start the learning process from the very beginning or use instead an already (partly) learned network (ResNet, YOLO). Pay attention to the size of the training set.
- If you design your own network, is it wise to use the raw images as input to the network (what will be the size of such a network? How many annotated real images should be necessary to train it?) or is it better to first do some preprocessing (downscaling, filtering, ...), feature extraction (edges, lines, ...), noise/outliers rejection, intensity normalization, ... to reduce the complexity of the problem and the dimension of the network?
- Obviously, the possible preprocessing and network architecture are different for the two image types.
- For both image types, any simplifying tricks or assumptions are welcome if you are able to argue your choice.

Results. In soccer images, you may view 0, 1, 2 or 3 ellipses in one image. In eye image, you may view 0 or 1 ellipse in one image (eye closed or open). You must detect all the ellipses, if any, which are present in an image.
There are three possible kinds of results of such modules;

1. **[Classification task]** The detection of ellipses in input images. You should choose/design and implement the criteria to decide if, and how many, ellipses are present in the image. This is a binary process which is similar to the detection of objects (like pedestrians) in images.
2. **[Regression task on bounding boxes]** The matching (regression) of the bounding box around the detected ellipses to approximatively locate them. Bounding boxes are often described by the position of the left, top, right and bottom sides of the rectangular box (4 parameters).
3. **[Regression task on ellipse parameters]** The matching (regression) of 5 geometric parameters of the detected ellipses. See below the next section on ellipse representation for more details.

Our suggestion is to implement preferably:

Tasks	Soccer images	Eye images
Classification task	X	X
Regression task on bounding boxes	X	
Regression task on ellipse parameters		X

Ellipse representation. Usually, we describe an ellipse by the 5 parameters (X_c, Y_c, θ, a, b), where;

- X_c and Y_c are the two coordinates of its center,
- θ is the orientation (angle) of its main axis and
- a and b are the half-length of its major and minor axis.

This parametrization is strictly equivalent to the representation of the ellipse by a rotated rectangle (as in OpenCV).
It is important to note that, when an ellipse tends to to a perfect circle, its axis half-lengths a and b tend to each other and its orientation θ becomes undefined (all orientations are equivalent for a circle). Other derived parameterizations exist but most of them experience some difficulties when the ellipse is almost a circle.
For instance, you may use the foci of the ellipse with the 5 parameters $(X_f^+, Y_f^+, X_f^-, Y_f^-, a)$, where;

- $X_f^\pm = X_c \pm c\cos\theta$ and $Y_f^\pm = Y_c \pm c\sin\theta$ are the coordinates of the two foci of the ellipse ($c = \sqrt{a^2 - b^2}$ is the distance between the ellipse center and one of its focus),
- and a is also the major axis half-length.

Another possibility is to use the bounding box of the ellipse with the 5 parameters $(X_c, Y_c, \delta_x, \delta_y, a)$ or (X_l, Y_t, X_r, Y_b, a), where

- X_c and Y_c are the two coordinates of its center,
- $\delta_x = \sqrt{a^2\cos^2\theta + b^2\sin^2\theta}$ and $\delta_y = \sqrt{b^2\cos^2\theta + a^2\sin^2\theta}$ are respectively the half-width and the half-height of the bounding box around the full ellipse,
- $X_l = X_c - \delta_x$, $Y_t = Y_c - \delta_y$, $X_r = X_c - \delta_x$ and $Y_b = Y_c + \delta_y$ are respectively the position of the left, top, right and bottom side of the rectangular bounding box around the full ellipse (with the axis origin located at the top left pixel of the image).
- and a is also the major axis half-length.

It is believed that these parameterizations based on the bounding boxes should demonstrate smoother behavior when the ellipse tend to a circle (in this case δ_x, δ_y and a tend to the same value). But pay attention to the fact that these parameters represent an ellipse if and only if the inequalities $\max\left(\delta_x, \delta_y\right) \leq a \leq \sqrt{\delta_x^2 + \delta_y^2}$ are verified.

Anyway, the best parameterization is always the one which matches your needs and fits well into your system!

Practically. Finally, you have to write and put into place two programs/modules (that follow the general guidelines described in section 3.1.1):

- They should take respectively a soccer or eye image as input.
- They must possibly preprocess the image before passing it to a trained network.
- The output of these modules are
  - the number of detected ellipses in an image, and
  - either the regression parameters of the detected ellipse, if any, around the pupil for the eye images
  - or the bounding box of the (1, 2 or 3) detected ellipses for the soccer image.
- You may choose to first detect (classification task) and then regress or the contrary or to do both at the same time.
- It is obviously suggested to use these numerical results to draw the detected ellipses or their bounding box on the input image to visually showcase the capabilities of the module.

3.1.5 Task 2.3: Image annotation

1. You will use the Cytomine software for the annotations. A presentation (tutorial) session of this Software will take place on Friday, the 15 of November after the theoretical course.
2. The images to annotate will be available in Cytomine on Friday, the 15 of November.
3. The annotations must be finished and available in the Cytomine software by Thursday, the 21 of November, no later than 23h29.
4. At first (from the 15/11 to the 21/11), each team have only access in Cytomine to the images it must annotate.

5. The number of images to annotate depends on the team size. For one person in a team, you must annotate;
   (a) the line segments of 5 images from the database provided for the first part,
   (b) the circles (1, 2, or 3) in 25 soccer images (you should NOT annotate the lines on these images, only the ellipses),
   (c) the pupil in 75 eyes images,
6. So, the groups with 4 (resp. 3 or 5) persons will annotate and provide;
   (a) the position of the line segments for 20 (resp. 15 or 25) images of the first part (in Cytomine, their name starts with 'line_', for instance 'line_soccer_0020.png' or 'line_sudoku_00033.png'),
   (b) the bounding box of the ellipses for 100 (resp. 75 or 125) soccer images (in Cytomine, their name starts with 'elps_soccer', for instance 'elps_soccer01_1053.png'),
   (c) the parameters of the ellipses for 300 (resp. 225 or 375) eye images (in Cytomine, their name starts with 'elps_eye', for instance 'elps_eye01_2014-11-26_08-49-31-060.png').
7. At the end of the annotation process (the 21 of November), the annotations will be shared by all the teams. Then, you will have access to
   (a) 850 annotated soccer images.
   (b) 2550 annotated eye images.
   (c) 195 annotated line images.
8. Ellipse annotations:
   (a) In Cytomine, you must annotate an ellipse with a polygon.
   (b) You must click at least 8 points (as spaced as possible) on the contour of the ellipse. Indeed, 5 points would be the very minimum to uniquely define an ellipse, but due to the limited precision of the position of the clicked points, it is explicitly requested to select at least 8 points.
   (c) When you get back the annotated point lists in your python code, you may use, for instance `cv2.FitEllipse` to obtain the geometrical parameters of the selected ellipse.
9. Bounding box annotations:
   (a) In Cytomine, you must annotate a bounding box with a rectangle.
   (b) you must click two opposite corners of the rectangle.
10. Line segment annotations:

(a) In Cytomine, you must annotate a line segment with a polyline.

(b) you must click the two end points of the line segment.

3.1.6 Task 2.4: Performance assessment of the ellipse matching module

This task is aimed at assessing the performances of the ellipse detection and matching module (see 3.1.4).

- It is a quantitative assessment, where you will compare the results of your modules with the ground truth on the annotated images.
- You are free to choose/design the comparison metrics which seem the most appropriate in this case.
- Practically, there are three quantitative assessments and you should measure the quality of;
  1. **[Classification task]** The detection of ellipses in input images.
  2. **[Regression task on bounding boxes]** The obtained position of the bounding boxes of the circles of the field marking in soccer images.
  3. **[Regression task on ellipse parameters]** The obtained parameters of the ellipses for the pupils in eye images.
- Practically, you will write programs to implement your metrics and perform the comparisons on all the images annotated by all the students.
- In your report, you will describe the metrics used and you will present the results obtained on these annotated images.

3.2 Presentation and reporting

The report is due by Thursday, the **12th of December**, no later than **23h59**, by mail sent to philippe.latour@uliege.be.
As explained in section 1.4.1, the report must contain:

- A `jupyter` notebook (NO PDF!) of maximum 15 pages (you should do a print preview of the notebook in your browser and look at the page number of the produced document). In addition to what is explained in 1.4.1, for the four modules, we expect:
  - A clear explanation of the assessment methodology and choices.
  - Your results and discussion concerning the performances of the module.

  - You may make reference to your first report, if needed.

- 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).

- Your code should not be entirely in the notebook. Instead, we expect you to import in the notebook the code you developed and that you want to demonstrate in the notebook.

3.3 Project presentation and test on theory

The presentation will take place on **Friday, the 20th of December starting at 8h00**, the room will be defined later.

It will be followed by the theoretical test (three teams at a time), the room will be defined later

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.

A preliminary schedule for the presentation and test on theory could be

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