Graduation Thesis
Annual Ditch Inspection
UAV Line Following

Author: Xin XU
Version 6.0
2015.6.4
Author: Xin Xu
Student No.: 69451
Company: NHL Centre of Expertise in Computer Vision
Place: Leeuwarden
Educational Institution: HZ University of Applied Sciences
Major: Mechatronics Engineering
Supervisor teacher: J.P.C Wubben
In-company mentor: Miquel Junyent Barbany
# Content

1 Introduction & Problem statement ........................................................................... 3
1.1 Problem Statement ................................................................................................. 3
1.2 Project Organization ............................................................................................... 3
1.3 The NHL Center of Expertise in Computer Vision ..................................................... 4
1.4 Research Background ............................................................................................ 6

2 Research/Development Method .............................................................................. 8
2.1 Iterative Method + Delft Design Method (DDM) ....................................................... 8
2.2 Version Control System ......................................................................................... 9

3 Theoretical Framework ............................................................................................. 10
3.1 Delft Design Method ............................................................................................. 10
3.2 Iterative Method ..................................................................................................... 10
3.3 The UAV ................................................................................................................ 10
3.4 PID Controller ....................................................................................................... 11
3.5 Computer Vision Software .................................................................................... 11

4 Analysis Phase .......................................................................................................... 12
4.1 Main Question & Sub-Questions ............................................................................. 12
4.2 Requirements and design specifications .................................................................. 13
4.3 Structure of the UAV ............................................................................................. 14
4.4 VisionLab .............................................................................................................. 16

5 Idea Phase .................................................................................................................. 18
5.1 Function tree ......................................................................................................... 18
5.2 Solutions ................................................................................................................ 19
5.3 Choice of two best Concepts .................................................................................. 20

6 Concept Phase .......................................................................................................... 21
6.1 Concept1 ................................................................................................................ 21
6.2 Concept2 ................................................................................................................ 23
6.3 Weighted Rating Matrix ......................................................................................... 26
6.4 Evaluation and Conclusion ..................................................................................... 27

7 Iteration Phase .......................................................................................................... 28
7.1 First Iteration ......................................................................................................... 28
7.2 Second Iteration ..................................................................................................... 35

8 Conclusion & Recommendations ............................................................................ 44
8.1 Conclusion related to the best Concept ................................................................... 44
8.2 Result for PID control .......................................................................................... 44
8.3 Recommendations .................................................................................................. 45

9 Discussion .................................................................................................................. 46

References .................................................................................................................... 47

Appendix ....................................................................................................................... 48
1 Introduction & Problem statement

Unmanned Aerial Vehicles (UAVs) are increasingly used for various applications, due to the great strides that have been put into their development of UAV. These applications range from entertainment and media to automation, and science. UAVs make it economical and easy to take images and other footage from the air. Some time-consuming activities that consist of inspecting or taking measurements can be performed by them. This makes companies be interested in using UAVs for their pursuits. A nice example shown in Figure 1.1 is a newly introduced application focuses on detecting young wild animals by means of heat detecting in order to protect them against running agriculture equipment (“De Telegraaf 30 mei 2015”).

![Figure 1.1 Detecting young animals with an infrared camera](image)

1.1 Problem Statement

Wetterskip Fryslân is one of the clients of the NHL Centre of Expertise in Computer Vision who wants to use a UAV to automate their processes, because the manual inspections appeared to be too costly. The Wetterskip wants to investigate whether it is possible and cost-effective to inspect ditches with drones instead of manually. It is necessary to check if the ditches are polluted and need to be cleaned or when they are cleaned, whether it is done properly. This is more cost-effective than doing it with manpower only. With the UAV, the water-management can be monitored more effectively and economically. By automating the ditch inspections the Wetterskip can inspect more important ditches more frequently. This will make the water-management more reliable and economical.

1.2 Project Organization

The project “ditch inspection” is divided in sub-projects to check if the complete concept is feasible. The sub-projects are:

a. Research if and how it is possible to detect vegetation in and around ditches and

b. Research the feasibility to enable a drone to follow a line with a downward camera.
In this graduation project, the feasibility of enabling an UAV to following a line is researched. The line represents a ditch that needs to be followed. This project will research the drone control possibilities for the ditch inspection. In order to inspect the ditches properly the drone has to follow the line at a certain speed and certain altitude. It can only use its own on-board systems to navigate. This means that it depends on its camera and image processing for flying over the line. A GPS navigation system will be not applied. The drone has to determine its position related to the line and correct itself when it moves away from the line. The research result will be applied in the future since this graduation project is a part of the main project.

1.3 The NHL Center of Expertise in Computer Vision

The NHL Center of Expertise in Computer Vision has many years of experience in applying computer vision for various purposes. It employs a fixed staff and varying pool of students (shown in Figure 1.2)

![Figure 1.2 The NHL Center of Expertise in Computer Vision](image)

The center of expertise is currently working on connecting UAVs with computer vision to improve or automate different tasks. The UAVs can be setup to perform different tasks with specific hardware. A prototype designed and built for automating inspections is shown in Figure 1.3.

![Figure 1.3 NHL prototype drone](image)
NHL Centre of Expertise in Computer Vision specializes in the automation of visual inspections. With the aid of computer vision, the information obtained from images taken by a camera can then be used to drive and amplify other processes with added value.

For many companies that make or process products, visual inspection or measurement is important. With the aid of Computer Vision, it is possible to make these inspections or measurements automated. This will contribute to a more flexible and economical production. Computer Vision is a combination of creative technologies. Through the use of camera and display technologies, the quality of products and services can be improved. For example, the inspection of potato using computer vision can dramatically reduce the waste of manpower, money and time most importantly. Moreover, this reduction of manpower, money and time will definitely leads to a more efficient production. Figure 1.4 shows the whole process of potato inspection.

The NHL Centre of Expertise in Computer Vision is an expert in the field of image processing and has a network of professional employees. The lab gives lectures, workshops and courses, has contacts with suppliers, universities and research institutes, publishes, participates as an exhibitor at fairs and organizes regular open days.

---

Figure 1.4 Potato inspection
1.4 Research Background

A ditch is a small to moderate depression created to channel water. It can be used to drain water from low-lying areas, alongside roadways or fields, or to channel water from a more distant source for plant irrigation. Figure 1.5 is a picture of a ditch. The inspection of ditches is currently done with manpower, which takes a large amount of time and therefore money. Now the drone is expected to inspect them automatically with image processing methods, it will save a lot of money.

This main project consists of two parts. The first one is to investigate if it is possible to follow a ditch. The second one is to investigate if it is possible to inspect the ditch, mainly the unexpected things in it, such as dead plants, branches, and leaves. The most important part to realize the first goal (following a ditch) is to enable a UAV fly following a line first. The assignment for the student is to see if it is possible to enable the drone fly following a line, realizing the first part. And the result of this project (line following) will hopefully make it possible to start the next phase of the main project. The structure of the project refers to Figure 1.6.
Figure 1.6 Structure of project
2 Research/Development Method

In this chapter, the research methodology applied on this project as well as changes of it is discussed. Also the tool for version control is introduced, which is essential for software engineering.

2.1 Iterative Method + Delft Design Method (DDM)

This project is mainly about software developing so engineering methodologies like DDM might not be suitable and efficient. Therefore, an Iterative and Incremental development method is mainly used. Nevertheless, several parts of DDM are still going to be applied in this case. This means that there isn’t a long term planning in the iteration, but the next phase is based on the results of the current one. The advantage of this combination is that there is no need to make a new planning if the requirements of the project change or if something turns out to be a bad idea. Everything learnt from the current phase and can be used in the next. The time of each iteration varies between 1 and 4 weeks. DDM still plays an important role in innovating ideas and evaluation. But since it is a code development based project and there is no such a real product at the end but a code and demo, some parts of DDM are not available in this case.

Thus, the most effective way is to combine Iterative Method with DDM. The materialization phase of DDM will be replaced by iteration phase of Iterative Method. Figure 2.1 shows the flow chart of Iterative Method.

![Iterative Method Flow Chart](image)
2.2 Version Control System

In software development, version control is the most important tool to be used. Version control is a system that records changes of a file or set of files over time thus people can recall specific versions later. GIT is the name of a Distributed Version Control System (DVCS). In a DVCS, clients don't just check out the latest snapshot of the files: they fully mirror the repository. Thus, if any server crashes, and these systems were collaborating with it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data. In this project, GIT is applied to control the version of code.
3 Theoretical Framework

In this chapter, the knowledge and ideas that will provide guidance to this project are discussed and the references about them are included as well.

3.1 Delft Design Method

The Delft Design Method (DDM) is a systematical method for engineering project, which is chosen as the main part for this project. The Delft design guide [1] presents an overview of product design approaches and methods. It gives examples and explains how to use the methods. These will be the important support when using Delft design method.

3.2 Iterative Method

Iterative Method is a systematical method for software engineering project, which is another important part of design method used in this assignment. Iterative methodology and designer training in human-computer interface design [2] demonstrates software design approaches and methods, which will definitely provide guidance in this project.

3.3 The UAV

An unmanned aerial vehicle (UAV), known in the mainstream as a drone and also referred to as an unpiloted aerial vehicle and a remotely piloted aircraft (RPA) by the International Civil Aviation Organization (ICAO), is an aircraft without a human pilot aboard. It is necessary to get the basic information of it. A mini unmanned aerial vehicle (UAV): system overview and image acquisition [3] describes all the details of the UAV, such as components, structure, and limits.

3.4 The camera system

The camera system used on the drone is Imaging Development System (IDS) uEye Industrial Camera. It is equipped with a megapixel level CMOS sensor, and available in images of format in monochrome, color. Its outstanding sensitivity to light and durability makes it suitable for operation on the drone. IDS uEye UI-241LE [4] shows an example of an IDS camera, giving parameters, functions and applications of it.
3.5 PID Controller

The operation of the drone highly depends on the performance of proportional-integral-derivative controller on it. A PID controller is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable. Thus the knowledge of PID is the most basic knowledge that can be used in this project. *Robust PID controller design for an UAV flight control system* [5] mainly shows how a PID controller works on an UAV flight system, which is almost the same case of this project. It will offer a lot of experience about PID controlling on an UAV. Figure 3.1 shows the working principle of a PID controller.

![Figure 3.1 Principle PID controller](image)

3.6 Computer Vision Software

The Computer Vision software is to be regarded as the core of this project, which obtains necessary information from images. It is really a challenge because it involves both programming and image processing theories and algorithms. VisionLab is the development environment as well as a great in this assignment because it provides plenty of operators so that there is no need to copy the source code of some kind of image processing. *Image processing algorithms for UAV” sense and avoid”* [6] also provides some examples of applying image processing on UAV flight.
4 Analysis Phase

In this chapter, the introduction and analysis about the software and hardware used in this project is demonstrated. In addition, two main parts of this chapter will be discussed at the beginning of it, the main research question & sub-questions and engineering design specifications respectively.

4.1 Main Question & Sub-Questions

In Table 4.1, the main and sub research questions are shown.

<table>
<thead>
<tr>
<th>Main Question</th>
<th>Sub-Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is it feasible to enable the UAV fly following a line automatically by computer vision?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analysis Phase</th>
<th>Idea Phase</th>
<th>Concept Phase</th>
<th>Iterative Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What’s the structure and principle of the hexacopter (DJI f550)?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>What’s the working principle of VisionLab?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>How to control a hexacopter by VisionLab?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>What are the basic functions and extensible functions of the scripts?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>What are the possible solutions to realize functions?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>What is the best concept to realize following line scripts?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>How to design and implement scripts that realize basic functions?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>How to test all the functions?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>What test result is needed?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>What are the recommended configuration of PID controller?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1 Main and sub-questions
4.2 Requirements and design specifications

In the following phases, all the design and concepts of the script should meet the requirements in this Engineering Design Specifications. These requirements are more about how the drone should move and its limits. There will be other requirements in chapter 8, but they are more about the details in scripts. These don’t conflict with each other, because the requirements here are of macro view and other requirements are of micro view.

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limit</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>The script should execute 12 times per second (83.3 ms every loop) at least to keep the continuous control of the drone.</td>
</tr>
<tr>
<td>2</td>
<td>The model of drone applied in this assignment has to be DJI f550, as designed and built in the previous projects.</td>
</tr>
<tr>
<td>3</td>
<td>The drone cannot fly lower than 0.5m or higher than 3m.</td>
</tr>
<tr>
<td>4</td>
<td>The flying speed of the drone is no more than 0.5 m/s, considering safety when testing.</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>The error margin of current speed calculated in the script should be no more than 10% with real speed to keep the accuracy of the flight.</td>
</tr>
<tr>
<td>6</td>
<td>The error margin of angle to target position calculated in the script should be no more than 5° with real angle to target position to keep the accuracy of the flight.</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>All variables can adjust themselves automatically when one or more initial variables are configured.</td>
</tr>
<tr>
<td>8</td>
<td>Main test values like speed and direction should be displayed while testing.</td>
</tr>
<tr>
<td>9</td>
<td>There should be 2 modes in the script, simulation mode which display all parameters and PID testing mode which doesn’t display. They can be switched easily.</td>
</tr>
<tr>
<td>10</td>
<td>The drone has to warn (beep etc.) when there occurs an error or a limit has exceeded.</td>
</tr>
<tr>
<td><strong>Layout</strong></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>The structure of scripts should be clear and understandable with necessary description for future developers.</td>
</tr>
</tbody>
</table>

Table 4.2 Engineering Design Specifications
4.3 Structure of the UAV

In this chapter, the DJI f550 model of hexacopter is described. It is intended to answer the relevant sub-questions.

4.3.1 Twirre Structure

1. What’s the structure and principle of the hexacopter (DJI f550)?

Twirre is a new architecture for UAV platform designed by J. van de Loosdrecht for automatic flight. Figure 5.1 shows the Twirre architecture. *Twirre: Architecture for autonomous mini-UAVs using interchangeable commodity components* [7] introduces more details about Twirre.

Two control circuits, manual in the left and automatic in the right, are alternated using a switch. The manual circuit is similar to the ones used in manually controlled drone, with the pilot intervention switch as the main difference. The circuit consists of a standard wirelessly connected transmitter/receiver pair. The servo signals pass from the receiver through the pilot intervention switch to the flight controller, which translates the stick inputs to motor inputs (standard servo pulse-width modulated signals - PWM). The automatic circuit consists of two basic components, with their peripherals. There is the processor board, where the cameras are connected, and the micro-controller, with its sensors. An application that runs in the process board generates stick outputs using the camera and sensors information. These go through the micro-controller, where some safety measures are applied, and finally are given to the flight controller through the pilot intervention switch. All the components that comprise the Twirre architecture are described in the following paragraphs.

![Twirre Architecture Diagram](image)

*Figure 4.1 the Twirre architecture*
4.3.2 Components

The Twirre architecture is built on a standard DJI Flame Wheel 550 (F550) hexacopter, which has a frame, six motors, electronic speed controllers (ESC) and propellers. The flight controller is a NAZA-M V2 flight controller, chosen for its modular design and the ability to use GPS-oriented flights if GPS module is installed.

The processor board is a Commell LS-37B mainboard with an Intel Core i7 3820QM processor, that has a clock speed of 2,7GHz and a maximum turbo frequency of 3,8 GHz. It has 4GB of DDR3 RAM and a standard Ubuntu (operating system) 13.10 distribution is running on it. This built-in computer makes it possible to perform image analysis on-board. This real-time processing makes automatic flight possible.

An industrial machine vision camera is used, which is mounted in the bottom of the drone. Bottom camera: IDS UI-1221LE-M-GL with a 12mm lens. This camera is used because it has a global shutter and it is suitable for image analysis applications with motion.

The micro-controller board used is a MicroWii Flight Controller based on an ATmega32U4 micro-controller, as shown in Figure 5.3. In this application is not used as a flight controller, but it has a gyro and accelerometer (MPU-6050), a magnetometer (HMC-5883L) and a barometer (MS5611-01BA03) which can be useful in automatic flight. An ultrasonic sensor (SRF08) has been connected to the micro-controller for accurate altitude measurements. This last component is used in the states hover, take off and land.
There is a switch to change between manual and automatic control. This controls a RC dual receiver controller designed for training RC pilots, manufactured by Assan. It switches between the receiver (manual) and the micro-controller (automatic). The transmitter is a Graupner MX-12 and the receiver a Graupner GR-12, both operating in the 2.4 GHz band.

Figure 4.4 Receiver and transmitter

4.4 VisionLab

This paragraph describes VisionLab, which is the main developing software used in this project. And in this chapter, it is intended to answer two sub-questions.

4.4.1 VisionLab Principle

2. What’s the working principle of VisionLab?

VisionLab is a development environment for image processing applications, which is highly extensible. You can add your own operators and camera interfaces. It can be easily used and implemented in existing software. VisionLab also has an open camera architecture and users can easily add their own cameras. Standard VisionLab supports several interfaces. It also supports a variety of vision algorithms including:

- Image analysis
- 2D camera calibration
- Edge detection
- Filters
- Geometric operations
- Image math
- Labeling and Blob Measurement
- Morphology
- Segmentation
- Color processing
- Barcode identification
- Transformation (Distance, Hough and FFT)
- Pattern matching
The system consists of 4 main parts:

1. **VisionLib** A library of more than 300 operators for image processing, interfaces to cameras and a neural networks for classification.
2. **VisionCmd** A command interpreter for scripts. The scripts can be used for easy and fast development of applications or prototypes.
3. **VisionServer** A shell with a network interface around VisionCmd. With VisionServer it is possible to communicate over the Internet with a remote VisionCmd. Scripts can be developed and tested on a PC and be executed on an intelligent camera.
4. **VisionClient** A graphical user interface with which it is possible to experiment in a comfortable way with the VisionLib library and to develop and test scripts. An online help is available.

VisionLib, VisionCmd, VisionServer are completely written in ANSI C++. This software is developed and tested under Windows, Linux and Android. The software runs on x86, x64, ARM and PowerPC processors. Ports have been made to FESTO SBOx and Matrix Vision Blue Cougar.

Because VisionLab has an open structure it is easy for a user to accommodate or extend the system to his needs.

### 4.4.2 Connection with drone

#### 3. How to control a hexacopter by VisionLab?

A PID controller is the most important part of automatic flight. Scripts in VisionLab mainly process images and get necessary information, the PID controller and a control command called “DR_CONTROL” were designed and implemented in previous UAV projects. The inputs of the PID controller are the set-point values for four kinds of movement of the drone (pitch/roll/yaw/gaz) and the error values of them. The “DR_CONTROL” operator in VisionLab will receive the PID input and output to the drone continuously, so that VisionLab scripts can control the drone with the help of the PID controller.
5 Idea Phase

In the idea phase the functional tree is made and for each function possible solutions are found. At least for integral ideas are derived by means of a morphological diagram. Two of them are chosen as best concept and will be taken into the next concept phase.

5.1 Function tree

4. What are basic functions and extensible functions of the scripts?

Although the final product of this project is a code, it should still realize different functions of controlling a drone fly following a line separately. Figure 6.1 shows the relationships between every function.

But not all functions are going to be realized at this case. Therefore, five basic functions are chosen to be designed first. Other two functions are regarded as extra functions, which can be added after proving all basic functions are feasible.
5.2 Solutions

During several brainstorm sessions for each function several ideas are created. In table the ideas are shown in Table 5.2.

5. What are the possible solutions to realize functions?

<table>
<thead>
<tr>
<th>Functions</th>
<th>Ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove interference</td>
<td>Remove different color object</td>
</tr>
<tr>
<td>Detect line</td>
<td>Sobel edge detection</td>
</tr>
<tr>
<td>Orient direction</td>
<td>Circle mask</td>
</tr>
<tr>
<td>Get current speed</td>
<td>Marks</td>
</tr>
<tr>
<td>Control PID</td>
<td>Position-orientated</td>
</tr>
</tbody>
</table>

Table 5.2 Solutions

Based on the five basic functions, several solutions are innovated for each function. As in Table 5.2, four ideas are created by connecting different ideas of different functions. There is a priority of choosing solution because not every solution is effective and efficient. The names and words for computer vision are explained in appendix 1.

For removing interference, removing different color and area object are better than adding filters or changing camera settings. Because filters and camera settings depend a lot on environment such as brightness, and they might not be feasible in another environment.

For detecting line, Sobel edge detection and adding line mask are the best two because they are not affected by environment very much, while color detection is. And Skeleton detection needs a long time to calculate.
For getting current speed, using marks are the best way because it is flexible and costs nothing. GPS is effective but only feasible an outdoor environment. Optical flow might be suitable but it is still not developed enough to be applied in this project.

For orienting and controlling PID, solutions are in same priority. Red idea and blue idea are selected as two ultimate concepts while green one and purple one are abandoned.

5.3 Choice of two best Concepts

Two main concepts are selected from four combinations of solutions. These two concepts will be described and evaluated in the next concept phase.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Solution 1</th>
<th>Solution 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove interference</td>
<td>Remove small-area object</td>
<td>Remove different color object</td>
</tr>
<tr>
<td>Detect line</td>
<td>Sobel edge detection</td>
<td>Line mask</td>
</tr>
<tr>
<td>Get target position</td>
<td>Circle mask</td>
<td>Line mask</td>
</tr>
<tr>
<td>Get current speed</td>
<td>Marks</td>
<td>GPS</td>
</tr>
<tr>
<td>PID controller</td>
<td>Speed-orientated</td>
<td>Position-orientated</td>
</tr>
</tbody>
</table>

Table 5.3 Two best concepts
6 Concept Phase

In the idea phase the functional tree is made and for each function possible solutions are found. Two of them are chosen as best concept. In the concept phase, two main concepts will be described, detailed and evaluated. Since they are both ideas of coding, a morphological chart is hard to be shown in this case. In the following paragraphs, ideas and principles of those two main concepts will be discussed and the best one will be continued.

The combination of those solutions will contribute to two main concept of this assignment, and they will be described in the following paragraphs as well as those solutions.

6.1 Concept1

Concept 1 of the line following script consists of:

- a. Remove different color object
- b. Line mask
- c. GPS
- d. Position-orientated

Description:

- a. Remove interference

  In this concept, our team will try to use a tape whose color is totally different from objects around it as a line. It is feasible to distinguish a different color object by using HSV format image taken by camera. In Figure 6.1, it shows the structure of HSV, Value means how bright the color is, Saturation means how colorful it is and Hue means which kind of color it is. So in this case we can use the value of Saturation to remove other objects if the tape is colorful enough and use the value of Hue if the tape’s color is completely different from other objects.

Figure 6.1 HSV format
b. Detect line

A line mask is used to realize the function of detecting line. A line mask is a binary image with complete white background (pixel value equals 0) and lines (pixel value equals 1).

When the interference of the origin image is removed (different color objects removed), the image (with only a solid line) is multiplied with the line mask. Because in a binary image, the value of pixels in white is 0 and the value of pixels in black is 1, so the result is an array of parts of the line, but if we connect the gravity centers of those parts, a binary line will be obtained. As shown in Figure 6.3.
c. Get target position
   When multiplied with the line mask, the origin image of tape will become many pieces of the line as shown in Figure 6.3. Then the piece in the middle (the piece in red) will be regarded as the center of the drone, the red dot in Figure 6.3. Therefore the 5th piece in the left or right side of the middle piece will be selected as the target position to go.

d. Get current speed
   A GPS system will be used in this concept, because it's easy to get an accurate position. It is mainly used to get the current position

e. PID controller
   In this concept, the input of PID control is position-orientated, which means the error value of the PID is based on the difference between target position and current position. Getting current position can be realized by GPS system and getting target position is realized by the line mask. The script can record the parameter from GPS and calculate the error value by comparing the current position with target position.

6.2 Concept2

Concept 2 of the line following script consists of:
   a. Remove small-area object
   b. Circle mask
   c. Marks
   d. Speed-orientated
Description:

a. Remove interference

The tape should be largest object appearing in the image, so interference can be removed by calculating the area of blobs in the image. Blob is a group of connected pixels in a binary image. Blob analysis will be applied in this concept. Blob is defined as binary large object, which consists of two or more connected pixels of value 1. And blob analysis can show the properties of blobs like area, center of gravity. Figure 6.4 gives an example of bacteria analysis by using blob analysis.

b. Detect line

Sobel edge detection are used to detect the line in this concept. It is an algorithm in Visionlab that checks for changes in contrast in the image. Normally used for detecting edges of objects.

In this case, once the edge of the tape is obtained, the hole surrounded by the edge of the tape and the border of the image can be filled to become a large blob so that other smaller objects can be distinguished using blob area analysis. See figure 6.5.
c. Get target position

A circle mask is applied in this concept to get the target position. After getting the line by using Sobel edge detection. The line will be multiplied with a circle mask, and the result is always two pieces of the circle. Blob analysis can be used here to get the coordinates of those two pieces. One of those two coordinates will be selected as the target position according to where the drone heads. See Figure 6.6.

Figure 6.6 Result of circle mask

d. Get current speed

Some marks are used in this concept as the reference object to get current speed. In this case, marks are small pieces of tapes which are placed in fixed distance from other marks to ensure only one mark appears in every image. The difference of position of those marks in two images will easily show the speed of the drone.
e. PID controller

The input of PID controller is speed-orientated in this concept, which means the error value of PID is based on difference of speed in x-axis and speed in y-axis. So every loop, the script only need to calculate the difference between current speed-x and speed-y with desired speed-x and speed-y. And it won’t need any information about position.

6.3 Weighted Rating Matrix

6. What is the best concept to enable a machine move following a line?

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Weighted Factor</th>
<th>Concept1</th>
<th>Concept2</th>
</tr>
</thead>
<tbody>
<tr>
<td>The script should execute 12 times per second (83.3 ms every loop) at least when operating.</td>
<td>5</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>The error margin of current speed calculated in the script should be no more than 10% with real speed.</td>
<td>3</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>The error margin of angle to target position calculated in the script should be no more than 5° with real angle to target position.</td>
<td>3</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>All variables can adjust themselves automatically when one or more initial variables change.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The model of drone applied in this assignment has to be DJI f550.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The structure of scripts should be clear and understandable with necessary description.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The flying speed of the drone is no more than 0.5 m/s.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main test value like speed and direction should be displayed on computer screen while testing.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>There should be 2 modes in the main script, debug mode and operation mode. They can be switched easily.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The drone has to beep when there is error value.</td>
<td>Must do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The script can be applied to other environment not only lab environment.</td>
<td>4</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>
Other objects in images can hardly influence the script. | 3 | 3 | 9 | 5 | 15
There is no limitation on flying speed and direction of the drone in the script. | 4 | 2 | 8 | 4 | 16
Total Score | 74 | 86

Table 6.7 Weighted Rating Matrix

### 6.4 Evaluation and Conclusion

For removing interference, concept1 removes objects of different colors. It can be easily influenced by the brightness of the environment and it is also possible to have the same color object in the image. While concept2 doesn't have the limit of brightness because it only removes small area objects. It can hardly happen that an object as big as the line appears in the image.

For detecting line, concept1 uses line mask and concept2 uses Sobel edge detection. It is apparent that line mask is not feasible when the line is vertical unless the line mask also rotates itself. In addition, once there is a little interference not removed, it will also affect the accuracy greatly. On the other hand, Sobel edge detection is much more effective. It is not influenced by brightness and direction.

For getting target position, a circle mask is better than a line mask because the line mask is limited to the direction of the line and the line mask gives a lot of points as result which may reduce the accuracy. While the circle mask only gives two coordinates as result and it is very accurate because the center of the circle is the center of the camera on the drone.

For getting current speed, a GPS system is far more accurate than some marks. But a GPS system will cost a large amount of money as well as a large portion of power on the drone. What's more, the GPS doesn't work under an indoor environment, which means it is not possible to test in the lab. So some marks is more suitable for a lab environment and it cost almost nothing but some tapes. Although it is not such accurate as GPS, it can also keep the error margin with in 10%.

For the input of PID controller, position-orientated input means a GPS system is necessary because it is impossible to get information about position without such system. But speed-orientated input means it only cares speed difference in x-axis and y-axis and it doesn't matter the position. In the other way, direction-orientated input eliminates several limitations that exist in position-orientated mode.

In conclusion, concept 2 is the best concept considering all five functions. But it is still need to be improved in the future that how to find another way to replace marks. It will be mentioned in the conclusion of this report.
7 Iteration Phase

Iteration phase replaces the materialization phase of DDM as mentioned in the choice of design methodology. In iteration phase, the whole process of planning, designing, testing and evaluating will be demonstrated. It has to be mentioned that in this phase, only basic functions are aimed to be realized, not including extra function such as stop when detecting end of line, find a line when no line in sight. However, those extra functions will be added in the next phase as long as all the basic functions are feasible.

7.1 First Iteration

7.1.1 Planning

Since the basic idea is decided in the concept phase, more possibility and improvement about the idea will be discovered in this phase. The whole script is divided into 4 important sub-scripts to realize 4 main functions, which is shown in Figure 7.1.

**Image to binary**: it processes the image taken from camera, distinguishes the line and speed marks from all the interference, creates two new binary images, one with the line only and one with speed marks only. It realizes functions of removing interferences and detecting the line.

**Orientation**: it searches for the binary image of line and get direction that the drone should head to. It realizes functions of getting target position.

**Speed**: it checks the binary image of speed marks, calculate speed on x-axis and y-axis by the movement of speed mark. It realizes functions of getting current speed.

**PID control**: it initializes and update PID controller. It realizes functions of controlling moves of the drone

![Figure 7.1 Workflow of scripts](image)
The structure of state machine is used in this iteration. A state machine is a mathematical model of computation used to design both computer programs, which conceived as an abstract machine that can be in one of a finite number of states. It is divided into one main state and two error-states. State 0 contains orientation script and speed script. State 1 and 2 are executed when line is out sight or speed is unknown. Theoretically, the state machine will always go to state 0 first to detect line in orientation script. If a line is detected, it will still go to state 0 next loop; if no line detected, it will go to state 1 next loop; if a line is detected but speed is known, it will go to state 2 next loop. Figure 7.2 shows the structure of the state machine of this project.
7.1.2 Requirements

The requirements listed in Table 7.3 are mentioned in chapter 4, which are about details and scripts, of micro view. Due to the research result, there may be several additional requirements, which need more researching, and testing.

| Camera setting | 1 | Format of image taken from camera is gray scale image. |
| 2 | Size of image is 1280 * 1024. |
| Image to binary | 3 | Input is the gray scale image from camera. |
| 4 | Output are two binary images of line and speed mark respectively. |
| 5 | It can remove all the interference effectively. |
| 6 | Width of the line in binary image should be 30 to 60 pixels. |
| Orientation | 7 | Input is the binary image of line. |
| 8 | Output is the angle(s) it finds, max 2 by principle. |
| 9 | Radius of search circle is 2 to 3 times the width of line. |
| 10 | It can detect if line is out sight or even no line. |
| Speed | 11 | Input is the binary image of speed mark. |
| 12 | Output are deviation from the set-point speeds on both x-axis and y-axis. |
| 13 | Positive speed means the drone moves forward, and negative speed means backward. |
| 14 | It can return error message like speed unknown or speed too high. |
| PID control | 15 | Input is error values of speeds on x-axis and y-axis. |
| 16 | Clamps should be set to avoid extreme high PID value. |
| Others | 17 | It can create a log file that records important parameter while operating. |

Table 7.3 Requirements for iteration phase
7.1.3 Implementation

Based on the idea from idea phase, four functional scripts and state machine are designed. The working principle is described in the following paragraph. In Appendix 1, there are more details and demonstration of pictures about implementation in Iteration 1st.

7. How to design and implement scripts that realize basic functions?

The following are the procedures of all scripts, which contains several VisionLab functions, “Sobel edge detection”, “Threshold”, “Fill holes”, “Remove blobs”, “Blob analysis”. They can be checked in VisionLab course pdf [8] easily.

a. **Image to binary**
1. Input gray scale image from camera.
2. Use Sobel edge detection to get the outline of line and speed mark.
3. Threshold the outline image to remove interference.
4. Use “fill holes” to fill speed mark and line is still only with outline.
5. Remove blobs which are bigger than some size, remove speed mark actually. Only the outline of line left.
6. Subtract the image with filled speed mark and outline of line by the image with only outline of line. Only speed mark left.

b. **Orientation**
1. Input the binary image only with the outline of line.
2. Multiply outline with the circle mask. if there are three or more intersection points, it will keep running; if there are two or less intersection points, it will return value of state 1 (line out sight).
3. Use blob analysis to analyze those intersection points. Get coordinates of them by checking center of gravity in blob analysis. Get final target point by averaging coordinates of the most top two points.
4. Calculate the angle between horizontal line and the line connected the center of image with the target point.
c. Speed
1. Input the binary image only with speed mark.

2. Use blob analysis to get the coordinate of speed mark by checking center of gravity.

3. Subtract the x-vector of the coordinate with it in the image of previous loop, and divided it with the time of the loop. Get the speed on x-axis.

4. Subtract the y-vector of the coordinate with it in the image of previous loop, and divided it with the time of the loop. Get the speed on y-axis.

d. PID control
1. Get set point value of speed on x-axis and y-axis by decompose desired speed with the angle get from orientation script.

2. Get error value of speed on x-axis by subtract set point speed on x-axis with real speed on x-axis.

3. Get error value of speed on y-axis by subtract set point speed on y-axis with real speed on y-axis.

4. Update PID controller by input the real time and error values.
7.1.4 Testing

8. How to test all the functions?

The tests for this iteration take place in the laboratory. In Figure 7.4, black tapes are put on the white wall as the line and speed marks. The camera is fixed on a frame so that it can move steady and can be kept at a fixed distance from the wall. To simulate the flight of the drone the frame is pushed slowly along the wall. All the important parameters and images are displayed on a screen. All the functions are tested separately first, and then all of them are tested together to see the overall performance.

![Figure 7.4 Test setup in the lab](image)

a. **Image to binary**

It works in a normal situation, where the brightness is distributed averagely and little interference. When the brightness of environment changes, it does not work. Because the outline of speed mark may not be closed, which leads to the failure of filling blobs and removing blobs. As a result, binary line image and binary speed mark image are not available at this case. Then the orientation and speed scripts will not work either. Thus it will go to the state of line out of sight when the line is in sight actually.

b. **Orientation**

The orientation script works pretty well wherever the line is located in the image, besides the problem caused by image to binary script. It takes a relatively long time (around 130ms per loop), because it has to create a new circle mask image in every loop. Also the blob analysis takes some extra time, because it checks every pixel in the image every time.
c. Speed

It works when there is only one speed mark in the image. But when the old mark is out of image and the new one is just in, there will be an extreme high speed because the script thinks the change of speed mark is the move of speed mark. That happens when there are two marks in the image too.

d. PID control

PID controller is unable to be tested because only simulation is in this iteration and it can only be tested after all other functions are proved to be feasible.

7.1.5 Evaluation of First Evaluation

After testing, the main functions of scripts are realized but still have limitations and disadvantages. Especially the image to binary script is easily influenced by the brightness of environment and it affects the quality of line binary image and speed mark binary image. The Sobel edge detection is a good way to detect lines. The concept of this part is effective, but some parts have to be improved or replaced by more dedicated functions.

Secondly, the orientation-script works smoothly under every situation, which proves that the idea of circle mask is feasible. Creating a new image and checking every pixel in every loop really wastes some time. So some way like upload the circle mask into the server on the drone can be discussed. The way of getting the coordinates of blobs should be improved considering it costs extra time as well.

Thirdly, speed mark functions well when there is only one speed-mark in the image. To avoid the extreme high speed, setting a clamp can be useful. When there are two marks appearing at the same time in the image, a way should be found to let the script focus on the previous one until it disappears.

In conclusion, all the ideas are proved to be feasible despite of some limits and bugs. Some possibilities are mentioned in the paragraph above and they will be covered in the next iteration.
7.2 Second Iteration

7.2.1 Planning and requirements

The same requirements are implied as the first iteration. But there are a few additional requirements in this iteration:

a. The script must be executed within 80 milliseconds for every loop in order to control the drone continuously;
b. The speed-script showed a lot of excessive high values, which needs to be prevented;
c. Important parameters have to be displayed when testing the drone;
d. The drone has to log all the important parameters for analysis;

The orientation script takes 130 milliseconds to execute, due to the use of a few heavy operators. This script alone exceeds the time requirement. There is no way to increase the computing power on the drone, so a faster way of detecting the line has to be found.

The speed script shows occasionally very high values. This problem has to be solved because the speed is used as the feedback for the control-loop. The cause of the problem lays in the use of undefined marks that are placed on the ground. There can be more than one mark in the image. The script does not detect and remember which mark it is following. When it switches to a different mark, the traveled distance is much higher than normal. Speed marks appear and disappear at time because the drone is moving. The biggest problem is that the script changes the speed mark more often than necessary.

For testing purposes with the drone the operator has to be seen in real-time how the scripts perform. This only has to be the most important values. Which value is important depends on a specific test.

After the test log files are desired to analyze the complete test. These files can contain a lot more information than the values that are shown real time.
7.2.2 Implementation

a. Image to binary
The script image to binary has to be changed due to the changes made in the orientation-script. The script had to output a solid line instead of the edges of the line. The edge detection turns out to be a good way to detect the line. By filling between the edges a solid line can be made. To fill in the edges the object has to be closed. The two lines have to be connected together at the borders of the image. A new operator is made by the center of expertise for this problem. The operator checks the border of the image for objects. If it detects an object, it will add a line on the border until it detects another object. This will close the edges of the line and make the image ready to fill in the line. This will take just a few milliseconds more. Figure 7.5a shows the image before “add partial border” while 7.5b shows the image after that.

![Figure 7.5a](image1.png) ![Figure 7.5b](image2.png)

Add partial border

b. Orientation script
To speed up the script has to be implemented different. The operators that are used now consume too much computing time. By optimization of the current script, the operation time of the new operator for orientation went down to 60 milliseconds. This is fast enough in principle, leaving just 20 milliseconds for the rest of the scripts. This is very minimal so the script has to be improved further.

Visionlab has the ability to add custom operators. These operators can be programmed in C++, which consumes less computing power than the Visionlab pseudo code. For this problem a new operator is made. This operator takes an image as input and radius as input and returns the angles were a line is found. This operator works on the same principle as the previous script, but without the Visionlab operators and with less image processing. It only analyzes the image and returns the result. The result of implementing the new operator is that the computing time of the orientation script went down to 18 milliseconds.
As shown in Figure 7.6, the operator “find circle intersections” needs line binary image and the coordinate of its search circle as well as its radius as input. Then it will return 2 coordinates as result, which should be the two points crossed with the search circle on the line. And the top one is what is needed for navigation. But to make the new operator work correctly, it needs a solid line as input.

c. Speed script

The speed measurement has to be more robust. The script changes speed-marks too much. This is caused by uniformity of the speed-marks. To overcome this problem the marks have to be different from each other and the difference has to be easy to detect for the script. This difference can be done by different colors, shapes or sizes or the drone could always use for example the left one. These options are all easy to recognize for the drone. The different colors are difficult to implement, because the rest of the scripts works on gray-scale images. The different shapes are easy to detect, but take a lot of computing time. Different sizes are easy to recognize and easy to implement. The necessary operator “blob analysis” is already used with the speed-marks, it only has to sort them on size. Always using the left speed-mark will cause the script to switch marks more than necessary, because it will switch to another mark every time when the most left mark disappears.
The difference in size is used to prevent the script from switching between speed-marks. There are two different marks used, small and big ones. There can be two speed-marks in the image without any problems.

The script still has to change the mark it is following from time to time. To prevent extreme speed peak a median filter is used. The median filter is a nonlinear digital filtering technique, used to remove the abnormal speed that is too high or too low. This will not affect the signal very much, but ignores the extreme spikes.

d. Real-time information

The most important data has to be shown to the operator in real-time. Which values have to be shown depends on what the operator wants to test. The values are shown via the command in VisionLab, an example is given in Figure 7.7, where values are displayed on the topleft.

![Figure 7.7 Real-time information](image)

e. Log file generation

All the values that are interesting for later analyze of the flight are stored. After the scripts are finished the array is written to a comma separated format file, which can be opened by Microsoft Excel.
7.2.3 Testing

9. What test result is needed?

All the tests in this iteration are the same as in the first iteration. But this time a log file is generated to analyze all the important parameters, and a screenshot video is taken as well for analyzing the reactions of scripts.

a. Orientation
Orientation script works properly. It detects the line and returns the angles correctly. The first angle is always the angle pointing forward. This angle is used for the navigation, which means that it is not possible to rotate more than 90 degrees.

b. Image to binary
The image to binary script outputs two images one with the speed marks and one with the solid line. The new operator that adds a partial border to connect the two edges of the line has some problems. When there are small object at the border of the image due to noise or unevenness of the background, the closing of the line goes wrong. This will result in a useless image.

The operator that adds the border to close the edges starts checking at the left top point of the image and proceeds clockwise. If the line is diagonal and the left top of the image is in between the two edges, the line will be inverted.

c. Speed
In speed script, the speed became a little less accurate, but within acceptable margins. The speed-mark following decreased the amount of extreme spikes in the speed significantly. The speed script has an absolute accuracy error less 8%. This is measured by moving the camera 1 meter and take the average speed and the total time. The total traveled distance it measured was always within 8% margin, which causes the same result in speed.

The x-axis returned an inverted speed. When the drone moves to the right, it actually sees the ground moving to the left. This was solved by inverting the output of the speed-script. This problem did not occur on the y-axis, because y-coordinates are already inverted in Visionlab images.

d. PID control
PID controller is unable to be tested because only simulation is in this iteration and it can only be tested after all other functions are proved to be feasible.
7.2.4 Evaluation of Second Iteration

After testing, it can be concluded that all the scripts work perfectly under most situations. But in some special situation when small object touches the border of the image, the image to binary script didn't work, because the new operator that adds a partial border to connect the two edges of the line has some problems. When there are small object at the border of the image due to noise or background, and when the line is diagonal and the left top of the image is in between the two edges, the closing of the line goes wrong. This will result in a useless image. Removing small object in the image before closing the edges may solve the problem. There is really low possibility when those two situations happen, therefore they can be ignored first until a better solution is found.

Orientation script is improved a lot, both the time it takes and its accuracy, due to the new operator. This script is already feasible to realize the function of orienting, it's perfect for this project.

Speed script always gives a constant error margin of 8% approximately, it can be solved by multiplying with another constant like 0.92. No extreme high value appears again when testing, but adding a filter will be a good idea to remove occasional abnormal values.

Now all the scripts except the PID controlling part are functional, so PID tuning and speed script improving will be the most important part in the next section.
7.3 PID testing

7.3.1 Test setup

In previous projects done with the drone, a position-oriented PID controller was used. In this project a position-oriented PID controller is not usable, because there is no position reference when following a line. A speed-oriented PID is used to control the drone’s movements. This is the first project within the Centre of Expertise that uses a speed-oriented PID controller. Previous experience with the PID controller of the drone cannot be used since it will react differently to the gains compared to the position-oriented PID controller.

The drone’s PID outputs result in proportional acceleration to the input of six motors on the drone. An input of roll will result in acceleration to the left or the right. The pitch has the same behavior, but will result in a forward and backward acceleration. The PID-controllers will control the amount of acceleration depending on the error value of speed.

The PID’s have to be tuned to control the drone properly. The procedure that will be used is to start with a proportional gain. This gain will be set low in at start and will be increased until an optimum gain is found. Than a derivative of an integral will be added and be tuned together with the proportional gain.

The PID controller of the drone will take inputs from -1 to 1. In order to prevent high acceleration, the input values are clamped at -0.5 and 0.5. This gives the pilot more time to take back the control and prevent crashes.

The PID tests take place in a room that is big enough for the drone to move, as shown in Figure 7.8. Also two boards with black tapes and marks on them are used as the line needs to be followed, the boards are shown in Figure 7.9.

![Figure 7.8 Testing room](image)
7.3.2 Test results

Only the test with the PID controlling rolling is finished due to the availability of the drone. Thus the test result is all about controlling rolling by PID. But in principle, Pitching and rolling should be the same configuration of PID controller, because they are both the results of the orientation script. Thus the PID controlling of pitching should be feasible when the PID controlling of rolling is feasible.

**Proportional gain**

First, it is started with a low proportional gain. This gain is determined by the following calculation:

\[
\frac{\text{maxValue}}{\text{maxError}} = P_{\text{gain}}
\]

\[
P_{\text{gain}} = \frac{0.5}{300} = 0.0017
\]

To be safe, the gain 0,001 is chosen. With this gain the controller did not react well on the error values. It lost the line very quickly due a very minimal reaction to the errors. In order to get a decent output from the controller, the P-gain had to be increased. The optimum on the P-gain was around 0,0035. A higher gain made the system unstable and a gain lower than 0,0035 made the controller underreact too much. With only a proportional gain the controller could not function properly.

Even with the optimum proportional gain it kept losing the line, due too much overshoot and not responding errors fast enough.
**Derivative gain**

A Derivative gives the controller the ability prevent overshoot as it reacts on a change of the error value. A derivative gain of 0,3 increased the controller performance significantly. With the P and D combined, the value of the proportional gain could be increased as the derivative gain prevents overshoot that caused by a high proportional gain.

The derivative part leads a new problem in the control-loop. The derivative part reacts on change of the error value, which causes it to amplify the high-frequency noise induced by the speed feedback. High frequency noise can be filtered out with a low pass filter. This filter makes the original signal pass through, but blocks the high-frequency noise. In Figure 7.7, the speed signal with noise is shown in blue. The speed signals processed by median filter and moving average are shown in green and blue respectively. The window sizes of those two filters are four. They both remove the high-frequency noise, although the moving average has more delay in the signal. Median filter with size four is chosen to reduce the noise in the signal.

![Figure 7.7 Filters for speed (x-axis is iteration time and y-axis is speed in mm/s)](image)

After implementing the median filter of size of four, both gains could be increased further. With the proportional gain 0,005 and the derivative gain of 0,4, the drone is able to control its position of left and right above the line by controlling rolling of PID controller. There is still some overreaction from the derivative part shown in the flight dynamics of the drone. This makes it react really aggressive on noise.

In Appendix 3, there are two graphs recording the behavior of the drone when testing PID, proportional gain of 0,005 and derivative gain of 0,4.
8 Conclusion & Recommendations

For the result of this research, it will be divided into three main parts: the result for concept, the result for PID controlling and recommendations respectively. In addition, since all the sub-questions are answered and the research is finished, the main research question can be answered. Until so far, it is not completely feasible to enable the UAV fly following a line automatically by computer vision. Because of the availability of the drone, only tests about rolling is done, which proves controlling the drone above the line on the x-axis is feasible. However, in theory, controlling the drone on y-axis should be feasible because it is the same principle as controlling on x-axis. Thus, there will be more tests about controlling it on the y-axis in the future.

8.1 Conclusion related to the best Concept

The best concept chosen from the concept phase is proved to be feasible through the implementation and testing of two iterations. Only basic functions are realized due to limited time, but extra functions are nothing more than the extensible part of these existing scripts.

8.2 Result for PID control

It is the first time for the NHL Centre of Expertise in Computer Vision try the speed-oriented PID control since the position-oriented one is proved feasible in previous UAV projects. For the speed-oriented PID tuning, it is hard to make the drone stay stable due to the continuous changing set-point value and error value. But still, the best configuration of controlling rolling is found by plenty times of testing. Therefore, the speed-oriented PID control is proved to be feasible as well and it will be an extra option for controlling PID for future UAV projects.
8.3 Recommendations

Last but not least, there are several recommendations and suggestions for continuing this assignment as well as the future project that refer to this research. First, the idea of speed mark can be replaced by the optical flow application in the future, because there should always be at least a speed mark near the line in this project, which is quite annoying if it is a long line.

Second, now only straight line with curves are tested. The behavior of the drone following other different shapes of lines, such as T-junction and 90-degree turning, is unknown. The suggestion is more tests on lines of different shapes in the future.

Third, a mathematical model of speed-oriented PID is suggested to be created. It is hard to set the model by the operation parameters of the drone, because the set-point value of PID controller is always changing. But it is possible to take a video of the drone when adding a step function for input and analyze the speed from the video. Thus a mathematical model might be created, which will be really useful for future research.

Fourth, further combination with GPS navigation-supported equipment might be taken in consideration.
9 Discussion

The result of this project is intended to help explore more possibilities of the automatic flight of the UAV supported by computer vision. The straight line following of the drone is realized in this project, thus more complicated routes might be followed hopefully in the future. Also, the speed-oriented PID control is first tried among all UAV projects. It provides guidance for different ways to control PID in the future. The weak point is that the PID tuning is nearly impossible to achieve by setting a mathematical model because the changing set-point value of it. In the future research, the PID tuning will be still only realized by configuring parameters of PID controller and checking the behavior of the drone, if there is still no method to get a mathematical model.

9.1 Self-reflection

In this project, I really expand my programming skills and have a basic command of computer vision through coding, debugging. In addition, I understand PID control more clearly, by observing the different behaviors of the drone of different PID configuration. Moreover, I’m familiar with the operation and organization of a company, working with colleagues makes me a communicative team player. Making report and delivering presentation also help me practice how to introduce my ideas more efficiently to other people.

By reviewing the whole process of this project, there are still several points need to be enhanced:

a. Time should be used more efficiently. For example, results should be recorded whenever there is one, so that there is no need to recall everything in the end.

b. Preparation for tests should be more comprehensive. Because there are two infeasible PID tests due to the failure of not fixing bugs in advance.

c. For innovation of computer vision, I should think more like a computer but not a human. Thus, the designing part will be more logical and reasonable for a computer vision program.
References


# Appendix 1

## List of Abbreviations

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobel edge detection</td>
<td>An algorithm in Visionlab that checks for changes in contrast in the image. Normally used for detecting edges of objects.</td>
</tr>
<tr>
<td>Blob</td>
<td>Group of connected pixels in a binary image</td>
</tr>
<tr>
<td>Git</td>
<td>Version control program used by the Centre of expertise</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle, an aircraft without a pilot onboard. Referred as drone in this report</td>
</tr>
<tr>
<td>PID controller</td>
<td>A control loop feedback mechanism. A mechanism that attempts to minimize the difference between the set-point and the measured value of a variable</td>
</tr>
<tr>
<td>Arduino</td>
<td>A micro-controller board</td>
</tr>
<tr>
<td>Flight controller</td>
<td>An onboard microcontroller that receives inputs likes left and right and translates this to the amount of throttle for each engine</td>
</tr>
<tr>
<td>DDM</td>
<td>Delft Design Methodology, is a systematical way of executing a project</td>
</tr>
<tr>
<td>Iterative method</td>
<td>A systematical way of executing a project, normally for software engineering</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation, is a technique used to encode a message into a pulsing signal</td>
</tr>
<tr>
<td>Mask</td>
<td>A preset set of numbers to multiply with every single pixel or the complete picture</td>
</tr>
<tr>
<td>Optical flow</td>
<td>Motion detection by comparing a frame with the previous one.</td>
</tr>
</tbody>
</table>
Appendix 2

Implementation of Iteration 1st

The following are the procedures of all scripts, which contains several VisionLab functions, “Sobel edge detection”, “Threshold”, “Fill holes”, ” Remove blobs”, “Blob analysis”. They can be checked in VisionLab course pdf [7] easily.

Image to binary
1. Input gray scale image from camera.

2. Use Sobel edge detection to get the outline of line and speed mark.
3. Threshold the outline image to remove interference.

4. Use “fill holes” to fill speed mark and line is still only with outline.
5. Remove blobs which are bigger than some size, remove speed mark actually. Only the outline of line left.

6. Subtract the image with filled speed mark and outline of line by the image with only outline of line. Only speed mark left.
Orientation

1. Input the binary image only with the outline of line.

2. Multiply outline with the circle mask. If there are three or more intersection points, it will keep running; if there are two or less intersection points, it will return value of state 1 (line out sight).

3. Use blob analysis to analyze those intersection points. Get coordinates of them by checking center of gravity in blob analysis. Get final target point by averaging coordinates of the most top two points.
4. Calculate the angle between horizontal line and the line connected the center of image with the target point.
Speed
1. Input the binary image only with speed mark.

2. Use blob analysis to get the coordinate of speed mark by checking center of gravity.

3. Subtract the x-vector of the coordinate with it in the image of previous loop, and divided it with the time of the loop. Get the speed on x-axis.
4. Subtract the y-vector of the coordinate with it in the image of previous loop, and divided it with the time of the loop. Get the speed on y-axis.

PID control
5. Get set point value of speed on x-axis and y-axis by decompose desired speed with the angle get from orientation script.
6. Get error value of speed on x-axis by subtract set point speed on x-axis with real speed on x-axis.
7. Get error value of speed on y-axis by subtract set point speed on y-axis with real speed on y-axis.
8. Update PID controller by input the real time and error values.
Appendix 3

Test results PID with different gain

Figure A3.1 Testing PID with proportional gain of 0.005 and derivative of 0.4

Figure A3.2 Testing PID with proportional gain of 0.005 and derivative of 0.4
Appendix 4

Example of coding

```plaintext
#take new image
snapshot IDSCam Int16Image 0 image_origin

#get time
  $current_time = MilliTime Stamp
  $time = $current_time - $previous_time
  $previous_time = $current_time

#sobel image_origin speedmark whatever GradientMagnitude 1 100
threshold speedmark 100 1000
removeblobs speedmark EightConnected Area 0 300 UseX
addpartialborder speedmark 0 1 1
fillholes speedmark EightConnected

copy speedmark binary
#remove line from image "speedmark"
removeblobs speedmark EightConnected Area 10000 9999999 UseX

#remove speedmarks form image with line
subtract binary speedmark

Figure A4.1 Code of image to binary

#pixel <-> millimeter conversion, distance[mm] = distance[nPixels] * height[mm] / conversionfactor[]
  $$x_{distance} = $$x_{distance} * $$mona_height / 2259
  $$y_{distance} = $$y_{distance} * $$mona_height / 2259

#distance -> speed, distance[mm] / time[ms] * 1000
  $$speed_x = $$x_{distance} / $$time * 1000
  $$speed_y = $$y_{distance} / $$time * 1000

  $$speed_x = round $$speed_x
  $$speed_y = round $$speed_y
  $$result = $$speed_x . $$speed_y
  return $$result

Figure A4.2 Code of transformation from pixel to millimeter
```
// search for intersections
$$result = f indc i rcl e i n ter s e c t i o n s \ image \ \$\$ x _ { \text { centre}} \ \$\$ y _ { \text { centre}} \ \$\$ c i r c l e r a d i u s$$

// INTERPRET RESULTS
// line out sight
if $$result \text{ empty}$$ then
  $$line\_insight = 0$$
  $$\text{angle1} = \text{unknown}$$
  $$\text{angle2} = \text{unknown}$$
else
  $$result\_size = \text{vertarray} \ result \ size$$
  if $$result\_size = 1$$ then
    $$line\_insight = 1$$
    $$\text{point1} = \text{remove\_first\_word} \ result$$
    $$\text{point1}_x = \text{get\_nth\_from\_vector} 1 \ \text{point1}$$
    $$\text{point1}_y = \text{get\_nth\_from\_vector} 2 \ \text{point1}$$
    $$\text{ito0}_x = \text{point1}_x - \text{x centre}$$
    $$\text{cos1} = \text{ito0}_x / \text{circ\_radius}$$
    $$\text{angle1} = \text{acos} \ \text{cos1}$$
    if $$\text{point1}_y > \text{y centre}$$ then
      $$\text{angle1} = 0 - \text{angle1}$$
  endif
  $$\text{angle2} = \text{unknown}$$

Figure A4.3 Code of orientation