NTNU

Norwegian University of
Science and Technology

Computer Vision Based Autonomous Panel Intervention for a Remotely Operation Vessel

Libo Xue

Marine Technology
Submission date: July 2018
Supervisor: Martin Ludvigsen, IMT

Norwegian University of Science and Technology
Department of Marine Technology

dedication (optional)

Summary

The Remotely Operated Underwater Vehicle (ROV) has been operating for many years. The use of ROV is increasing with the increased activities in sub-sea missions which are moving towards deep sea. Therefore, the need to improve the autonomy of ROV is becoming more and more urgent. The purpose of this thesis is to develop a system that can allow this ROV to adjust its pose towards intervention objects and to locate the intervention targets.

The common used sensor for ROV intervention is the camera. And vast amount of research and tools already accessible from the field of computer vision. Therefore, it is advantageous to develop a system that utilizes these cameras.

This paper compares and analyzes previous research projects on underwater autonomous intervention, and analyze whether this thesis can use the methods of previous projects. The conclusion is new strategies and methods are needed for this thesis, when dealing with a 3D subsea module installed in real underwater environment.

A close-range visual guidance algorithm was proposed for ROV posture adjustment. This method utilize the line feature extracted from the camera image. The decision making of ROV motion is based on linear perspective theory. Besides, a label detection algorithm is developed as the starting point of valve intervention. These two algorithms are test with video record, the test result is in line with our expectations.

Field trials can be conducted in the future, in order to test the cooperation of algorithms and control systems. There are many works left for the valve intervention, for example the label recognition, valve panel SLAM and manipulator path planning. Besides, due to the existence of marine snow, autonomous dust cleaning may be a interesting topic which is also needed for ROV autonomous intervention architecture.

Preface

This thesis is based on the research carried out during the spring semester of 2018 at the Department of Marine Technology, Norwegian University of Science and Technology. This is a Master's thesis regarding ROVs autonomous intervention and is a result of both individual work and help from fellow students and professors at NTNU. The main topic of this thesis is to implement computer vision based guidance and object detection algorithm to improve the autonomous level of ROVs intervention mission.

It is assumed that the reader of this thesis retains a basic knowledge within engineering science.

Acknowledgements

I would like to thank my supervisor Prof. Martin Ludvigsen for giving me this opportunity to study this interesting topic. He provided me with feedback during the project period. He also gave me space to develop myself independently, which allow me learn a lot of new knowledge.

Further, I want to thank, Ph.D. candidate Stein M. Nordnes, M.Sc. student Chiu Jack, M.Sc. student Einar Agdestein and M.Sc. student Erik Holven for the academic discussions and help during the work of this thesis.

I am grateful to the M.Sc. student Chiu Jack for the motivating dialogues concerning the thesis, as well as in deciding upon the conversion requirement from ”Sonar Tracking” to the ”Camera Tracking”.

I want to thank Adrian Rosebrock for sharing his knowledge on his computer vision blog http://www.pyimagesearch.com/; it has helped me throughout the development of algorithms.

I would like to thank Ph.D. candidate Kaiming He, his paper ”Single Image Haze Removal Using Dark Channel Prior” inspired me to jump out of the thinking pattern. And I want to thank Boxian Chen, master of art at Tsinghua University, for his detailed and professional explanation of linear perspective theory.

I also want to thank the open source community of OpenCV, which has shared open source algorithms and tutorials on how one can implement computer vision methods in python. The OpenCV community has provided open source algorithms and packages vital for the success of this thesis.

Finally, I would like to thank my family for the support during my master program study.

Libo Xue

Table of Contents

Summary i

Preface ii

Acknowledgements iii

Table of Contents vii

List of Figures x

Abbreviations xi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Previous Projects . . . 3
  - 1.2.1 Underwater Autonomy . . . 3
  - 1.2.2 Underwater Visual Guidance . . . 4
- 1.3 Challenges . . . 6
  - 1.3.1 Marine Snow . . . 6
  - 1.3.2 Subsea Module . . . 7
  - 1.3.3 Valve . . . 9
  - 1.3.4 Illumination Condition . . . 10
- 1.4 Scope of Work . . . 11
  - 1.4.1 Infrastructure of this project . . . 11
  - 1.4.2 Challenges Focused on . . . 12
- 1.5 Contribution . . . 13
- 1.6 Structure of Thesis . . . 13

2 Theory 15
- 2.1 Digital Image . . . 16
- 2.2 Image enhancement in spatial space . . . 20

2.2.1 Gray level transformation . . . 20
2.2.2 Histogram Equalization . . . 20
2.2.3 Gray level slicing . . . 21
2.3 Morphology image processing . . . 21
2.3.1 Erosion . . . 22
2.3.2 Dilation . . . 22
2.3.3 Closing . . . 22
2.4 Image Segmentation . . . 23
2.4.1 Canny edge detector . . . 23
2.4.2 Hough Transformation . . . 24
2.5 Epipolar geometry . . . 26
2.6 Visual Guidance . . . 28
2.6.1 Feature descriptor . . . 28
2.6.2 Guidance strategy . . . 28
2.6.3 Linear Perspective Theory . . . 30
2.6.4 Vision-system based motion decision making . . . 34
2.6.5 ROV pose adjustment . . . 36
2.6.6 Line Classification . . . 37

3 Method **41**
3.1 Close Range Visual Guidance . . . 41
3.1.1 Degree of freedom reduction . . . 41
3.1.2 Scenarios . . . 42
3.1.3 Algorithm procedure . . . 44
3.2 Label Detection . . . 50
3.2.1 Strategy of valve detection . . . 50
3.2.2 Algorithm procedure . . . 50

4 Result **57**
4.1 Close Range Visual Guidance result . . . 58
4.2 Label Detection . . . 60
4.2.1 Label detection result . . . 60
4.2.2 Bad illumination . . . 60

5 Analysis and Discussion **61**
5.1 Uncertainties and Obstacle . . . 61
5.1.1 Field experiment . . . 61
5.1.2 Dusty surface . . . 61
5.2 Pros and cons of the methods . . . 61

6 Conclusion and future work **63**
6.1 Conclusion . . . 63
6.2 Further work . . . 63
6.2.1 Marine Snow . . . 63
6.2.2 Label recognition . . . 63

Bibliography **65**

Appendix **69**

List of Figures

1.1 Heavy wetsuit.(https://mypics.at/bilder/barcelona/maritim-museum/) . . . 1
1.2 Plan of ROV intervention mission.(Rist-Christensen, 2016) 2
1.3 landmark detection and keypoints matching.(Palomeras et al., 2016) . . . 4
1.4 Feature extraction results: (a) Line feature (b) corner and blob feature.(Palmer et al., 2009) 5
1.5 Image of panel surface (a) Jun 27, 2017 (b) Sep 21, 2017. 6
1.6 ORB features extraction result for a clean panel surface. 7
1.7 CAD model of the simplified 2D panel.(Palmer et al., 2009) 7
1.8 CAD model of the subsea module (3D) used in this thesis. 8
1.9 Line feature extraction result and intersections. 8
1.10 Illustration of valve and connector used in (Palomeras Rovira et al., 2016). 9
1.11 CAD of the real subsea module. 11
1.12 Minerva II. 12

2.1 RGB model cube.(Wikipedia contributors, 2018h) 16
2.2 (a) HSI model (b) HSV model.(Wikipedia contributors, 2018f) 17
2.3 Some basic gray-level transformation functions used for image enhancement.(Gonzalez and Wintz, 2010) 18
2.4 Plots of the gamma transformation grayscale conversion formula with various values of gamma (c = 1 in all cases).(Gonzalez and Wintz, 2010) . . 19
2.5 Gray level slicing.(Gonzalez and Wintz, 2010) 21
2.6 (a) Original image, (b) Erosion result. 22
2.7 (a) Original image, (b) Dilation result. 23
2.8 Canny edge detector results with different threshold values. (a) Original grayscale image (b) Canny result with c1 = 3 and c2 = 60 (c) Canny result c1 = 3 with c2 = 40 (d) Canny result c1 = 2 with c2 = 20 24
2.9 Dual relation between Cartesian cordinate and polar coordinate.Cartesian coordinate (left) polar coordinate (right). 25

2.10 The three points on a straight line in the Cartesian coordinate system correspond to the three curves in the polar coordinate which intersect at one point. . . . 25
2.11 Canny edge detection result. . . . 26
2.12 Epipolar geometry. . . . 27
2.13 Aircraft visual landing. . . . 29
2.14 Camera image of the subsea module. . . . 29
2.15 One-point perspective. . . . 31
2.16 Two-point perspective. . . . 32
2.17 Three-point perspective. . . . 32
2.18 Illustration of ”Y” type line classification. . . . 33
2.19 Case A. . . . 34
2.20 Case B. . . . 35
2.21 Image of corridor. . . . 35
2.22 The ROV rotation path. . . . 36
2.23 1st type of lines in the camera image. . . . 37
2.24 2nd type of lines in the camera image. . . . 38
2.25 3rd type of lines in the camera image. . . . 38
2.26 4th type of lines in the camera image. . . . 39
2.27 5th type of lines in the camera image. . . . 39
2.28 6th type of lines in the camera image. . . . 40

3.1 Division of area around the panel (top view). . . . 42
3.2 Scenario 1: only one side can be detected. . . . 43
3.3 Scenario 2: both two sides can be detected. . . . 43
3.4 Raw image capture by the camera. . . . 44
3.5 Mask created for raw image. . . . 45
3.6 Output of Canny edge detector. . . . 45
3.7 Canny edge detection result after using mask. . . . 46
3.8 Hough Transformation result comparison: CLAHE (left), HSV(right) . . . 47
3.9 Saturation channel of the camera image. . . . 48
3.10 Value channel of the camera image. . . . 48
3.11 Removal of duplicate lines. . . . 49
3.12 Shadow mode analysis of image. . . . 51
3.13 Remove the shadow from grayscale image. . . . 52
3.14 Histogram equalization. . . . 53
3.15 Otsu’s thresholding. . . . 54
3.16 Contours of detected labels. . . . 56

4.1 Video used for testing close-range visual guidance algorithm. . . . 57
4.2 Video used for testing label detection algorithm. . . . 57
4.3 Rotation angle and direction 58
4.4 Horizontal movment direction 59
4.5 Label detection with contour filter 60

Abbreviations

ROV	=	Remotly Operated Vehicle
DVL	=	Doppler Velocity Log
OpenCV	=	Open Source Computer Vision
RGB	=	Red Green Blue
HSI	=	Hue Saturation Intensity
HSV	=	Hue Saturation Value

Chapter 1

Introduction

1.1 Motivation

Early exploration of the shallow sea was carried out by human divers. To resist water pressure, divers need to wear heavy wetsuits and breathe through a tube.

Figure 1.1: Heavy wetsuit.(https://mypics.at/bilder/barcelona/maritim-museum/)

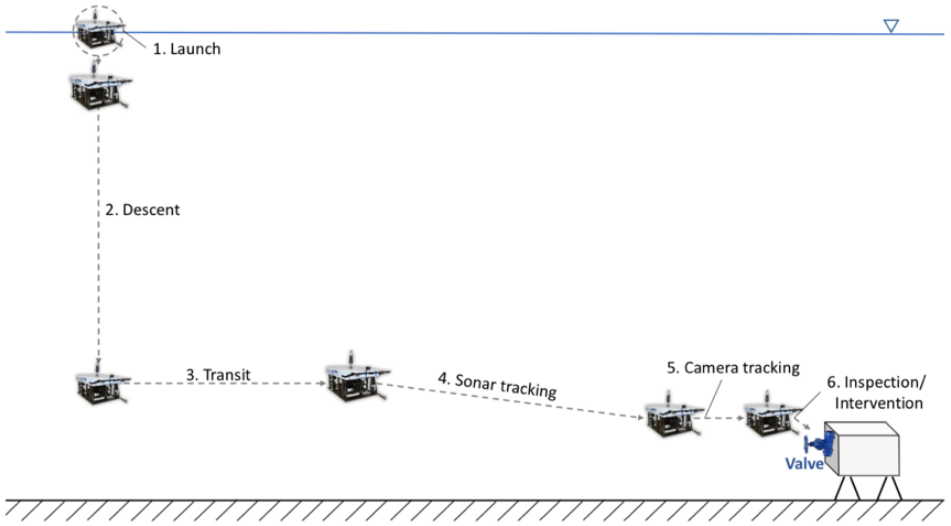

Figure 1.2: Plan of ROV intervention mission.(Rist-Christensen, 2016)

As the exploration depth of the ocean increasing, wetsuits can't protect divers from high pressure, and heavy diving suits severely limit their ability to move. Therefore, a variety of UUVs were invented to replace divers, such ROV, AUV. ROVs can be equipped with amount of sensors and robotic arms to perform a variety of tasks compared to AUVs, and can continue to receive power through the umbilical. However, ROV has its drawbacks compared with AUV. It needs surface support vessel and human drivers to perform tasks. The severe weather condition and high cost of operation limit the use of ROV.

To get rid of this limitation, removing the ship and people from control loop is a feasible approach. One idea is to increase the intervention ability based on the strong autonomy of the AUV, such as installing a robotic arm.(Youakim et al., 2017) Another way of thinking is to improve the autonomy on the basis of ROV's strong intervention ability. This thesis is based on improving autonomy level of a ROV.

Figure 1.2 shows the six steps of an ROV for underwater intervention.(Rist-Christensen, 2016) In order to remove people from this process, a vision-based obstacle avoidance algorithm was developed(Brusletto, 2016). This paper is dedicated to improving the autonomy of intervention detection, including close-range visual navigation and intervention target detection.

1.2 Previous Projects

1.2.1 Underwater Autonomy

For many years in the past, many research teams and corresponding projects have contributed to improving the autonomy of underwater robots. Some projects in the 1990s, such as OTTER(Wang et al., 1995), ODIN(Choi et al., 1994), UNION(Rigaud et al., 1998), AMADEUS(Lane et al., 1997), SWIMMR(Evans et al., 2001) and HROV(Fletcher et al., 2008)(Farr et al., 2010), made pioneering contributions to improving the autonomy of underwater robots. However, these projects did not conduct field experiments and some of them are still have human in the control loop.

Subsequent research projects have further breakthroughs. In 2003, the ALIVE project (Evans et al., 2003) firstly achieved the fully automated intervention. The panel was modified for the purpose of easy autonomous docking. After docking, fixed-base intervention was implemented using visual feedback. And in 2009, the SAUVIM project(Marani et al., 2009) firstly achieved free-floating manipulation. The robot successfully recover the object endowed with artificial landmarks. Then in 2012 TRIDENT project(Sanz et al., 2012) firstly proposed multipurpose object search and recovery strategy. Several field experiments with different manipulators were implemented consequently. Finally, in 2016, the PANDORA projec(Carrera et al., 2015) implemented free-floating valve-tuning on a sea panel. A low-cost SLAM was developed for object localization, and the Learning by Demonstration paradigm was used for valve-tuning.

We can summary from the above projects that there are two main strategies for underwater intervention: fixed-based and free-floating. The first one, fixed-based intervention, usually need to modify the panel to adapt the robot autonomous docking, such as the ALIVE project. The relative position between the robot and the object is limited to a certain range once the robot has docked. Then computer vision-related methods can be used to locate the intervention objects. The second one is free-floating intervention. This strategy has no need of panel modification, but artificial landmarks are required for intervention object, which will aid for object localization and detection, such as SAUVIM and TRDIENT projects. The introduction of SLAM and learning technology can reduce the reliance on both panel modification and artificial landmarks, as demonstrated in the PANDORA project achieves.

The fixed-based strategy indeed improves the operation reliability, but the requirement of panel modification will reduce the versatility, especially for the subsea equipment in service. Therefore, the strategy chose in this thesis is free-floating intervention. As mentioned above, the requirement of object detection will arise while utilizing the free-floating strategy. Considering that endowing artificial marks on panel will also result in the decrease of versatility, new method is needed. The SLAM algorithm introduced in PANDORA is a good one, which can localization and mapping at the same time. Once the map is obtained, it can be used for objects detection. However, this powerful algorithm requires extremely high computational cost. Furthermore, the poor underwater image quality will influence the quality of dense maps obtained by SLAM. Hence this thesis developed a lightweight algorithm, which is insensitive to image quality. Based on the linear perspective principle,

a progressive attitude adjustment algorithm was developed. In addition, based on shadow analysis method, the panel label detection algorithm is implemented. Valves can be located through corresponding panel label, eliminating the need of extra artificial landmarks.

1.2.2 Underwater Visual Guidance

Map-based localization and guidance

Accurate localization and guidance are crucial the robots using a free-floating strategy for intervention. The system will enable the robot to navigate in unstructured environments and locate desired intervention area and intervention objects. Considering a simple scenario, a precise map of the area is given, then mission will become navigating and localizing the robot in a structured environment.(Maurelli et al., 2008) Map-based method can simplify the mission but reduce the flexibility of the algorithm usage, since it cannot cope with the changes in objects or the environment.

SLAM

Compared to the less flexible map-based method, SLAM is a method which can solve both localization and guidance at the same time. Visual SLAM method can produce maps of the unstructured environment using image captured by camera, and the pose of robot can be calculated based on a series of frames. Several SLAM strategies for the underwater condition have already been proposed over the past few years.(Carrera et al., 2015)(Ribas et al., 2010)(Salvi et al., 2008)(Nagappa et al., 2013)(Ozog and Eustice, 2014) However, some of these strategies are difficult to implement in real-time due to the heavy computation. This is because the entire underwater environment is treated as unknown in these strategies. While the PANDORA project introduce prior knowledge of the panel into the system, reducing the amount of computation and enabling the real-time operation.(Palomeras et al., 2016) In the PANDORA project, the panel detection is based on ORB feature matching between panel template and image taken by camera. If there are sufficient matching features, robot pose and motion can also be estimated. The algorithm developed in PANDORA project was tested on a vertical panel surface placed in a water tank, shown in Figure 1.3, and artificial waves were added to simulate the ocean environment.

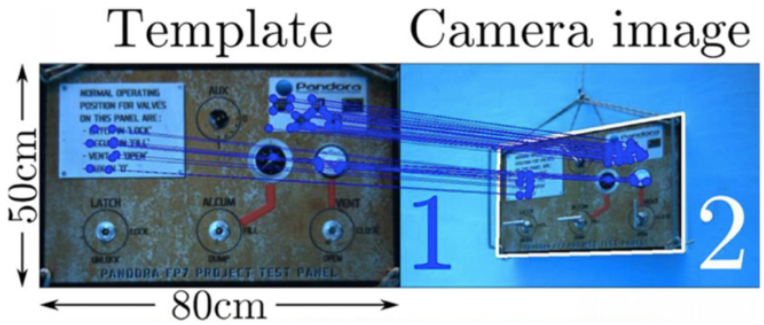

Figure 1.3: landmark detection and keypoints matching.(Palomeras et al., 2016)

Model-based localization and guidance

Model-based localization and guidance method utilize the detected geometric features to estimate the pose and motion. The key point of this method is find out representative feature points. Figure 1.4 (a) shows a vertically installed 2D panel plane. The center points of the valve and the connector are representative which can be extracted by blob detection. Besides, the intersecting points of the structure edges found by Harris corner detector are also useful.(Palmer et al., 2009) Figure 1.4 illustrate the feature extraction result. The Figure 1.4 (a) shows the line feature extraction result, and the Figure 1.4 (b) shows the corner and blob feature extraction result, where the black points are center points and red points are vertexes. Then the pose and motion of robot can be estimated based on Epipolar geometry (detail explanation will be given in Chapter 2).

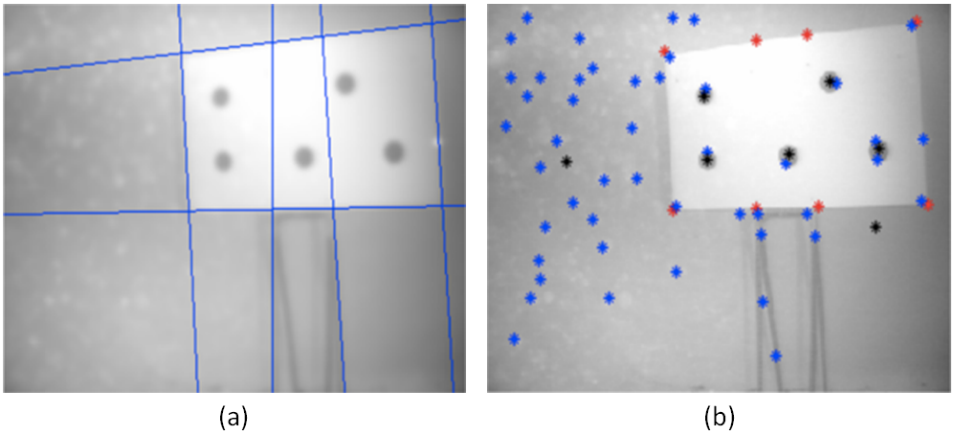

Figure 1.4: Feature extraction results: (a) Line feature (b) corner and blob feature.(Palmer et al., 2009)

1.3 Challenges

1.3.1 Marine Snow

Marine snow is an organic debris that settles like a snowflake in the deep sea. When light travels through the water and encounters these debris, the direction of propagation changes, creating a scattering phenomenon. This can cause black and white spots in the underwater image, which degrades the image quality. In addition, these debris will gradually precipitate over the surface of the underwater module, which will have an impact on vision-based feature extraction.

In (Carrera et al., 2015), the experiment was carried out in water tank. Although the influence of dynamics on navigation and localization was included, the impact caused by marine snow was not considered. The scattering will blur the underwater image which can influence the landmark detection and keypoints matching. Besides that the surface features will change as covered by sediments, which is crucial for landmark detection. Figure 1.5 (a) is taken on Jun 27, 2017, when the module used in this thesis was mounted on the Trondheimsford. Its horizontal top surface, are clean and free of dirt. While Figure 1.5 (b) is photographed on September 21, 2017. After 3 months placed underwater, the top surface is covered by a thin layer of sediments.

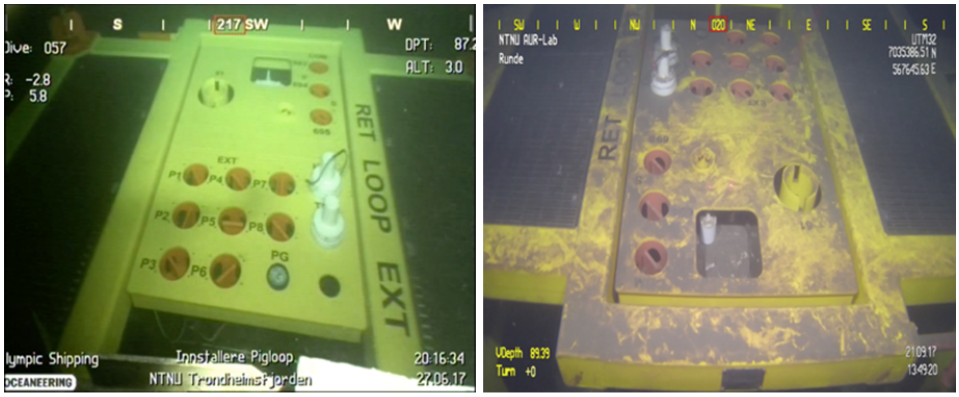

Figure 1.5: Image of panel surface (a) Jun 27, 2017 (b) Sep 21, 2017.

Figure 1.6 shows the results by using ORB feature extractor on Figure 1.5. The keypoints extracted is oriented FAST. As for a clean surface, as shown in Figure1.6 (a), ORB keypoints are mainly located at panel label and edges of structure. But the dust changes the keypoints distribution, as shown in Figure 1.6 (b), many error keypoints extracted from the edge of dust. This will definitely influence the keypoints matching between template and camera image shown in Figure 1.3. Hence, the SLAM method is not robust under this condition.

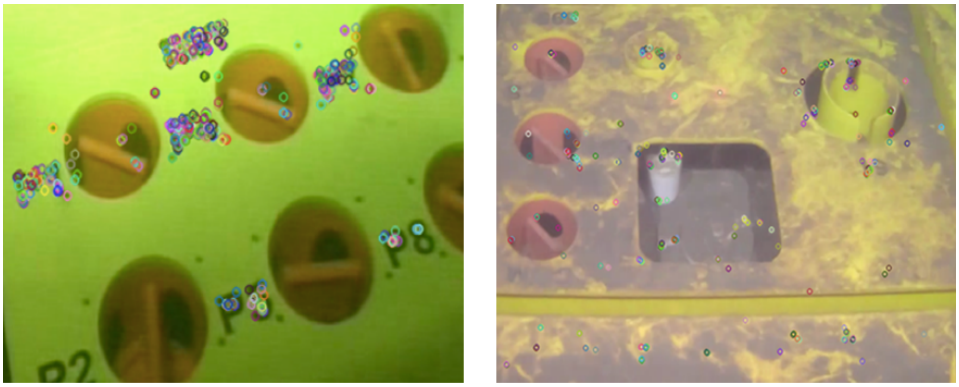

Figure 1.6: ORB features extraction result for a clean panel surface.

1.3.2 Subsea Module

The panel used in (Palomeras et al., 2016) is a simplified 2D plane vertically installed on the seabed, no other artificial markers or labels on it, only valves and connectors are kept as shown in Figure 1.7. Because only this plane exist in the space, all detected geometric features are belong to it as shown in Figure 1.4. Hence, pose and motion estimation based on the panel geometric features is easy to implement. The pose and motion estimation method can refer to Section 2.5.

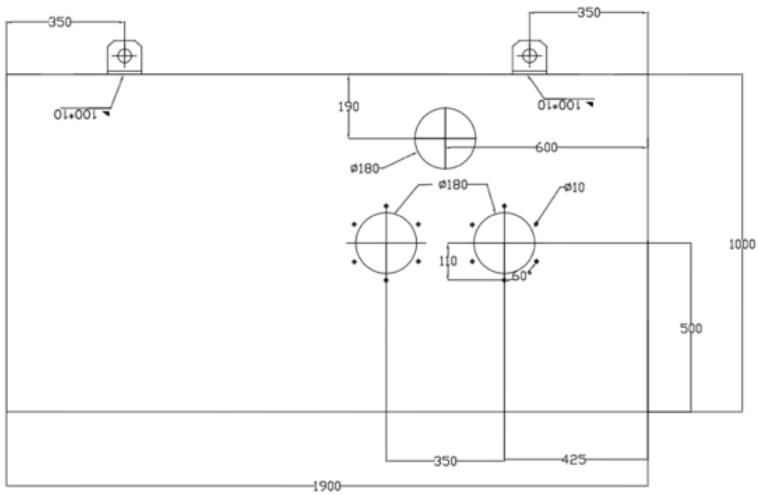

Figure 1.7: CAD model of the simplified 2D panel.(Palmer et al., 2009)

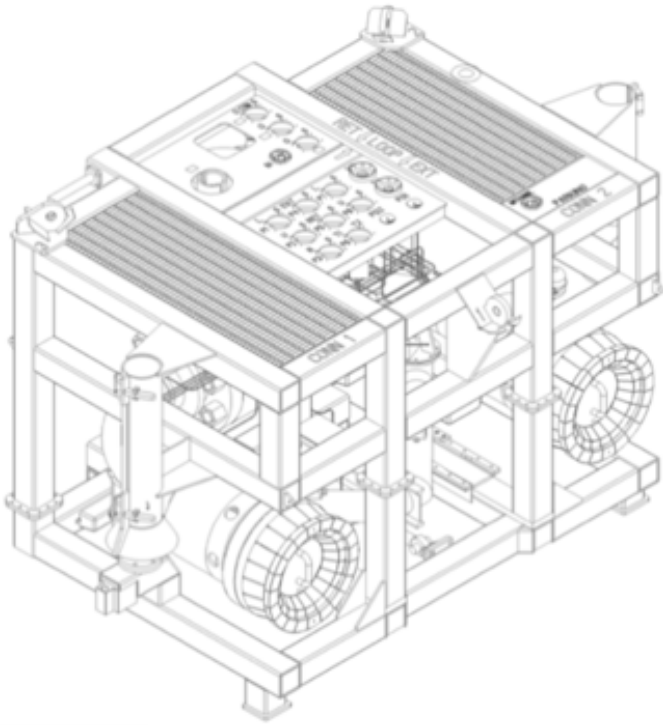

Figure 1.8: CAD model of the subsea module (3D) used in this thesis.

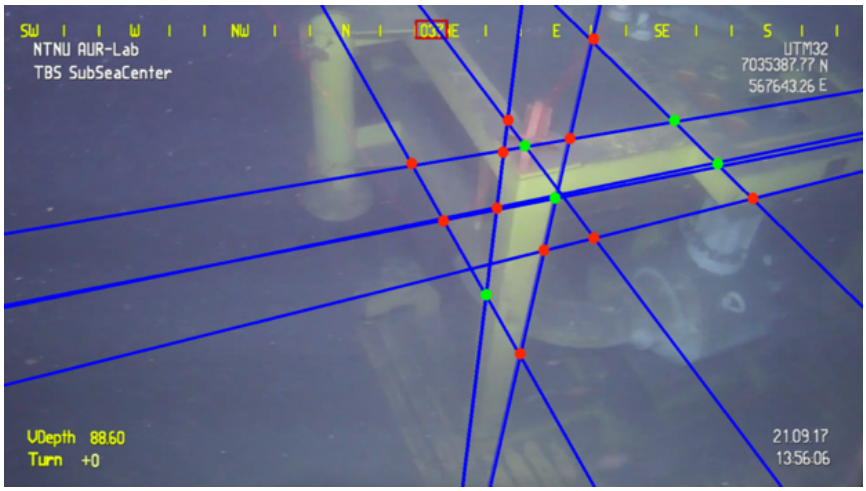

Figure 1.9: Line feature extraction result and intersections.

But situation is different when dealing with a real 3D subsea module as shown in Figure 1.8. There are many labels and auxiliary structures on it, which will interfere the keypoints extraction, for example Harris corner detection and Blob detection. Besides, due to the 3D structure, line features extracted may not belong to one plane which means intersections of projections of these extracted lines cannot represent geometric feature of the same plane. Figure 1.9 illustrates this. Intersections of extracted lines are highlighted by different colors. Green points represent true structural corners on the module. Red points are actually meaningless but will be regard as corner candidates by the Model-based method. This will result in wrong pose and motion estimation.

Hence, corner and blob are not feasible features used for model-based method when dealing with a real 3D subsea module.

1.3.3 Valve

In (Palomeras Rovira et al., 2016), fixed base intervention was carried out. Once the AUV docked, the valve will show up in the camera. This valve was modified to T-shape and red dots are added to three ends as shown in the Figure 1.10. The modification makes the valve to be easily detected by color. In (Palmer et al., 2009), based on the panel has been detected, valves can be positioned according to the simple asymmetric distribution on the panel as shown in Figure 1.7.

Because the number of valves is too large and the underwater illumination is not good, it is very difficult to give each valve a different color and successfully detect it based on color. This modification also does not apply to underwater modules that are in service. In addition, as explained in Section 1.3.2, the 3D subsea module increase the detection difficulty of individual plane panel. Therefore, valve detection based on panel detection result is difficult to achieve. In order to solve the valve detection for the module used in this paper, a new strategy needs to be proposed.

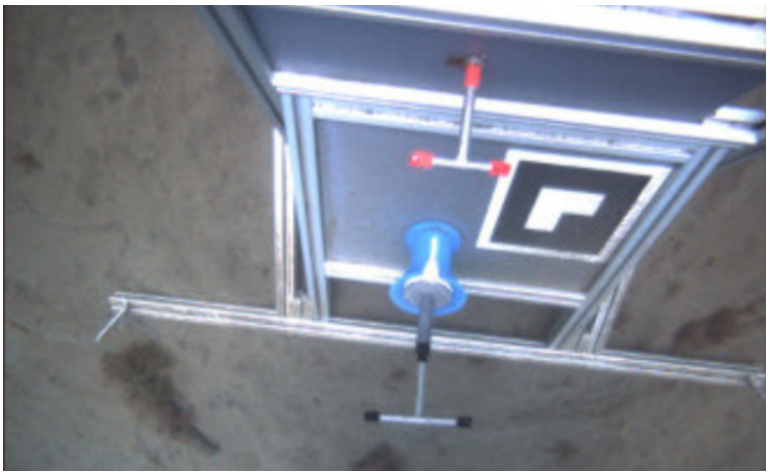

Figure 1.10: Illustration of valve and connector used in (Palomeras Rovira et al., 2016).

1.3.4 Illumination Condition

The underwater environment is dark due to the lack of sunlight. In order for the human pilot to be able to see the underwater environment, a searchlight is installed on the ROV. The searchlight is a point source, which causes uneven illumination distribution of underwater images. Because humans have a stronger ability to understand images, the effects of uneven illumination are not always obvious. But when computer dealing with the same situation, the effects of unven illumination will become critical. Computer vision understand a image based on basic image information, for example color, grayscale features and feature descriptors calculated etc. These image information are severely affected by illumination. Therefore, it is necessary to propose methods for appropriately removing the illumination unevenness for specific problems.

1.4 Scope of Work

In Section 1.2, previous projects are introduced and analysis is given in Section 1.3. Then, considering the specific intervention structure and underwater environment in this thesis, new strategy and method for underwater autonomous intervention need to be proposed and implemented.

1.4.1 Infrastructure of this project

Subsea module

The intervention target of this thesis is a real subsea panel, which is the intervention target of this thesis. There are 11 valves located on the top surface, which are enclosed in red dash line. The dimension of this panel is shown in Figure 1.11, in which the panel height (3020 mm) is an important threshold value for the ROVs altitude. This real subsea module is a cuboid module. The main structure lines can be categorized into three class of orthogonal lines, one class of lines is perpendicular to the seabed, and the other two class of lines are parallel to the seabed.

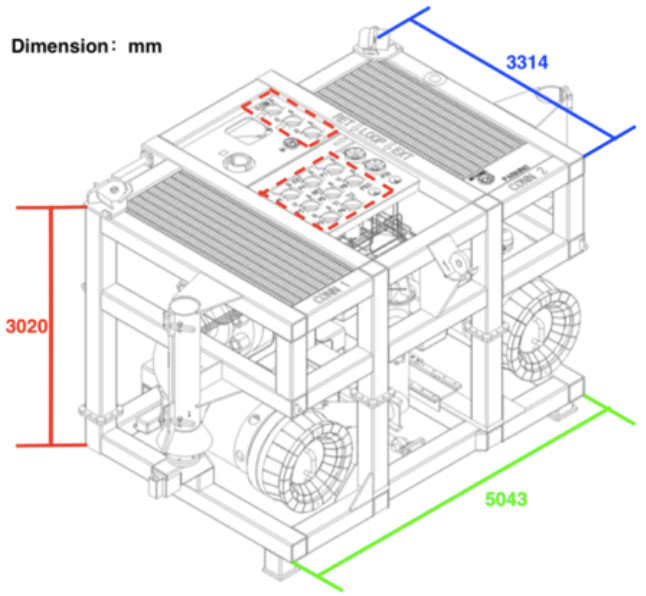

Figure 1.11: CAD of the real subsea module.

Minerva II

Figure 1.12 shows the ROV used in this thesis, which is equipped with cameras and Doppler Velocity Log. The dimension of it is $LWH = 290 * 170 * 160$.

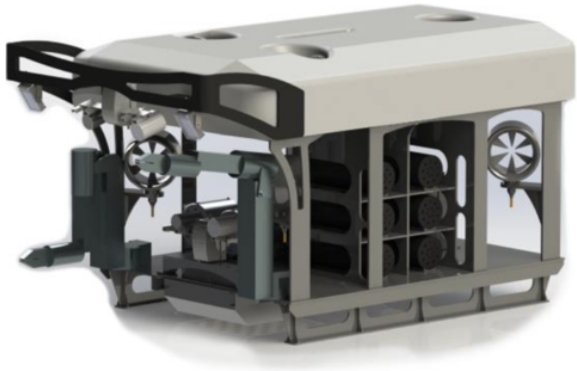

Figure 1.12: Minerva II.

1.4.2 Challenges Focused on

Although the pioneers have made many contributions to underwater autonomous intervention, considering the actual underwater environment and the real subsea module faced in this thesis, there are still many problems to be solved. This thesis will address some of the challenges discussed in Section 1.3. It should be mentioned that marine snow can affect image quality in two ways. In general, the scattering phenomenon caused by marine snow is considered as noise in the image and can be removed by a filter. Structural surface deposits caused by marine snow will directly alter surface features. In this case, it is impossible to rely on image processing techniques to recover surface information, but a surface cleaning operation is required. Therefore, this article does not directly propose a solution to the marine snow, but will consider its impact when solving other problems.

Subsea Module

Guidance based on corner and blob features are feasible for the 2D simplified panel used in (Palmer et al., 2009) but not the same for this thesis. To achieve the visual guidance, a new feature descriptor is proposed for the real 3D subsea module.

Valve

Because this thesis focuses on a real subsea module, it is impossible to simplify the module or modify the valve. Therefore, we need to find a way to combine the information provided by the labels and auxiliary structures on this real subsea module for valve detection.

Illumination Condition

Uneven illumination can degrade the quality of the image, which has a serious impact on image feature extraction. Therefore, in this thesis, special attention will be paid to solving the problem of uneven illumination when considering visual guidance and valve detection.

1.5 Contribution

The contribution of this thesis can be summarized as follow:

1. A close-range visual guidance algorithm is developed. This algorithm takes advantage of the line features of the structure and enables guidance with the help of linear perspective principles.

2. A valve detection strategy was proposed based on the real subsea module. And a valve detection algorithm based on shadow mode analysis has been developed.

1.6 Structure of Thesis

Chapter 1 introduction of this thesis.

Chapter 2 introduces theories relate to close-range visual guidance method and label detection algorithm.

Chapter 3 describes and illustrates the methods developed in this thesis.

Chapter 4 present the video test result.

Chapter 5 discuss and analysis results in the Chapter 4.

Chapter 6 concludes the thesis and suggests further works.

Appendix A Python source code.

Chapter 2

Theory

This thesis aims to use computer vision to increase the autonomy level of ROV. The traditional process of solving problems using computer vision is (Gonzalez and Wintz, 2010):

i. Image acquisition

ii. Image preprocessing

iii.Feature extraction

iv. Modeling

v. Output

This thesis does not focus on the methods and processes of image acquisition, but rather the acquired images, as it is the initial input to computer vision techniques. Hence the basics of digital image was introduced instead of image acquisition procedure in Section 2.1.

Sections 2.2-2.4 mainly introduce the processing effects and usage conditions of the image processing and feature extraction methods used in this thesis. Only necessary mathematical expressions and functions are given instead of detailed theoretical derivation. This is because the focus of this thesis is how to use these techniques and results obtained. As for the specific implementation principles and mathematical derivations can refer to (Gonzalez and Wintz, 2010).

Section 2.5 introduce the Epipolar Geometry which is the basic of pose and motion estimation. Both SLAM and Model-based methods utilize it to estimate the pose of ROV. Similar to Section 2.2-2.4, brief introduction of Epipolar Geometry and necessary mathematical expressions are given. For detailed introduction and mathematical derivations can refer to (Hartley and Zisserman, 2003).

Section 2.6 introduce the background materials relate to visual guidance based on Linear Perspective.

2.1 Digital Image

Digital image is the numeric representation of 2D image in computer. Three types of digital image are introduced in this thesis.

Binary image contains only white and black, which means it just has 2 gray levels. Usually some image processing algorithm use this type of image as input or output.(Wikipedia contributors, 2018d) Morphological processing is a commonly used method for processing binary images and will be introduced in Section 2.3.

Grayscale image contains 256 level of gray which make it able to present more complex scene than binary image.(Wikipedia contributors, 2018e) Many traditional image processing methods are developed based on gray level, such as image enhancement, restoration, etc.

Due to the improvement of hardware, color image becomes the commonly used form of digital image and we can extract more information from image content and colors. Digital image use color model to describe colors. It is the subspace of one specific coordinate system, in which colors are described by the coordinate of points. There are two types color model, one is hardware-oriented model and the other is visual perception model.

Hardware-oriented models

The RGB model is a hardware-oriented model commonly used in color monitors and cameras. This model is based on a Cartesian coordinate system, in which the coordinate of points is composed of three component values of red, green and blue.(Wikipedia contributors, 2018h) Red, green and blue are three primary colors of which values are in the range of (0,256). The RGB image can be converted into grayscale image by using Equation 2.7.

$$Grayscale = 0.299 * R + 0.587 * G + 0.114 * B \tag{2.1}$$

in which, R, G, B represent the values of corresponding channels in RGB image.

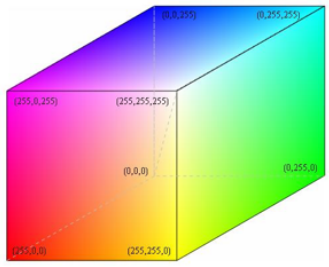

Figure 2.1: RGB model cube.(Wikipedia contributors, 2018h)

It is convenient for hardware to describe colors accurately using RGB model. But it is difficult for humans to directly determine colors from corresponding RGB coordinates because of the way humans understand colors. Hence, visual perception models are introduced to aid humans in development of computer vision algorithm.

Visual perception models

Both HSI and HSV are visual perception model based on cylindrical coordinate system, in which hue and saturation are used to describe colors.(Wikipedia contributors, 2018f) Hue is used to describe the solid color property, of which the value is in the range (0 - 360°). In which the 0° represents red, counterclockwise 120° is green, and 240° is blue. Saturation is a measure of the degree to which a solid color is diluted by white, of which the value is in the range of (0-100).The lower the saturation value, the higher the degree to which the solid color is diluted, which means the greater the proportion of white (S=0 is pure white).

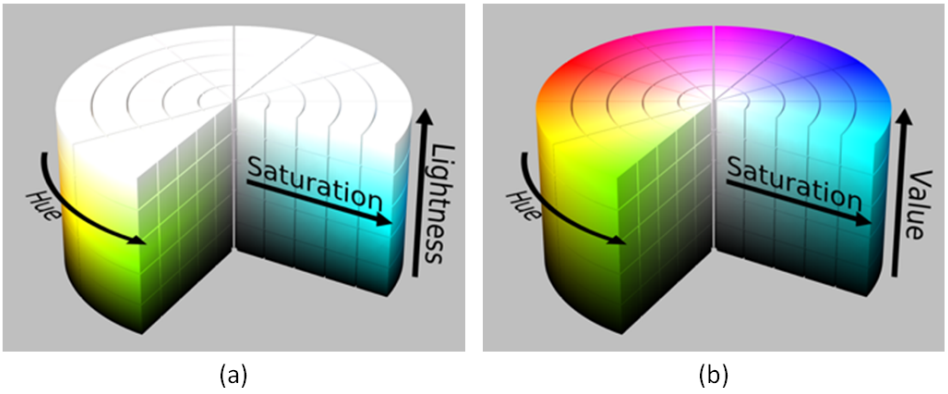

Figure 2.2: (a) HSI model (b) HSV model.(Wikipedia contributors, 2018f)

The HSI model is based on the human visual system, which is sensitive to brightness. But brightness is an intuitive descriptor and is actually unmeasurable, instead the light intensity is measurable. Hence the intensity is chosen as the third component in HSI model.

Although the HSI model is consistent with the way the human visual system describes color, it is different from the way the human brain does it. The method of color description by the human brain can be summarized through the process of painter color matching. The hue is clearly defined and constant, and more colors can be obtained by adjusting the color density (saturation) and depth (value). The color density is changed by adding white to hue, which corresponds to the concept of saturation. Color depth is changed by adding black to hue, which corresponds to the value concept. Hence value is chosen as the third component in HSV model.

In order to facilitate developer analysis and reader understanding, this thesis chooses HSV model to describe color image. To do this, the RGB image captured by camera need to be convert into HSV image. The conversion function in OpenCV library can be used directly

to achieve this transformation. The function is:

$$img = cv2.cvtColor(src, cv2.COLOR_BGR2HSV) \tag{2.2}$$

in which img is the image described in HSV color model, src is the original image. It should be note that in OpenCV library, the RGB image is stored in BGR mode, so using BGR instead of RGB when calling this function.

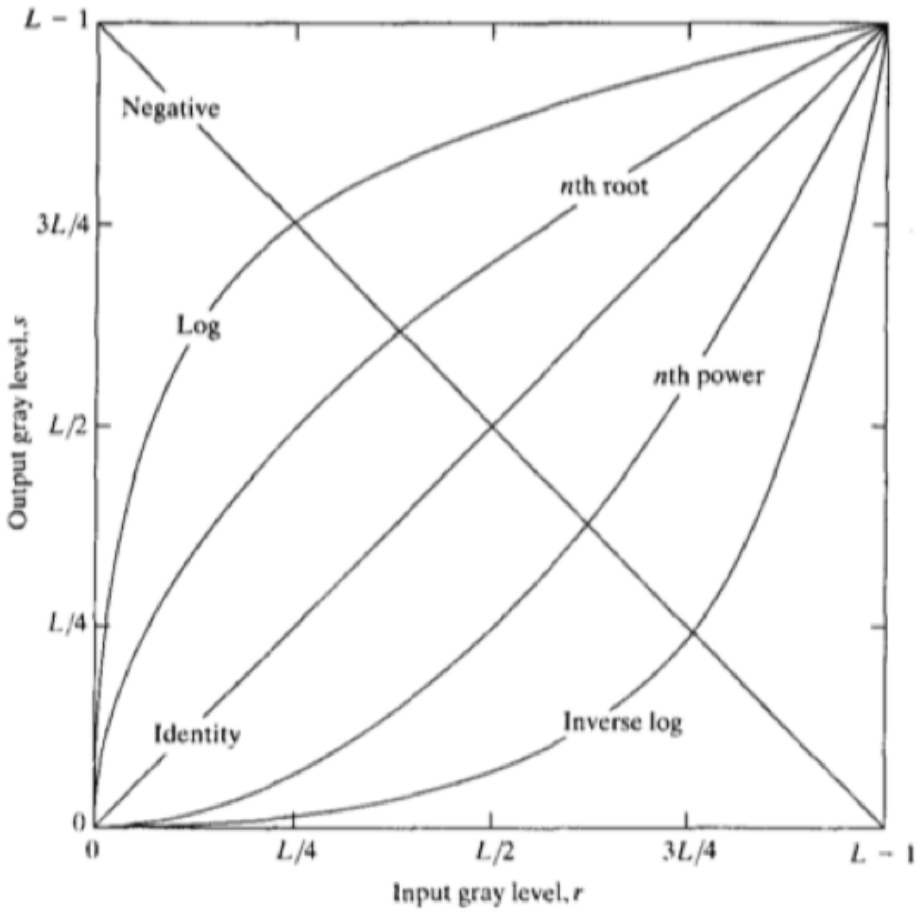

Figure 2.3: Some basic gray-level transformation functions used for image enhancement.(Gonzalez and Wintz, 2010)

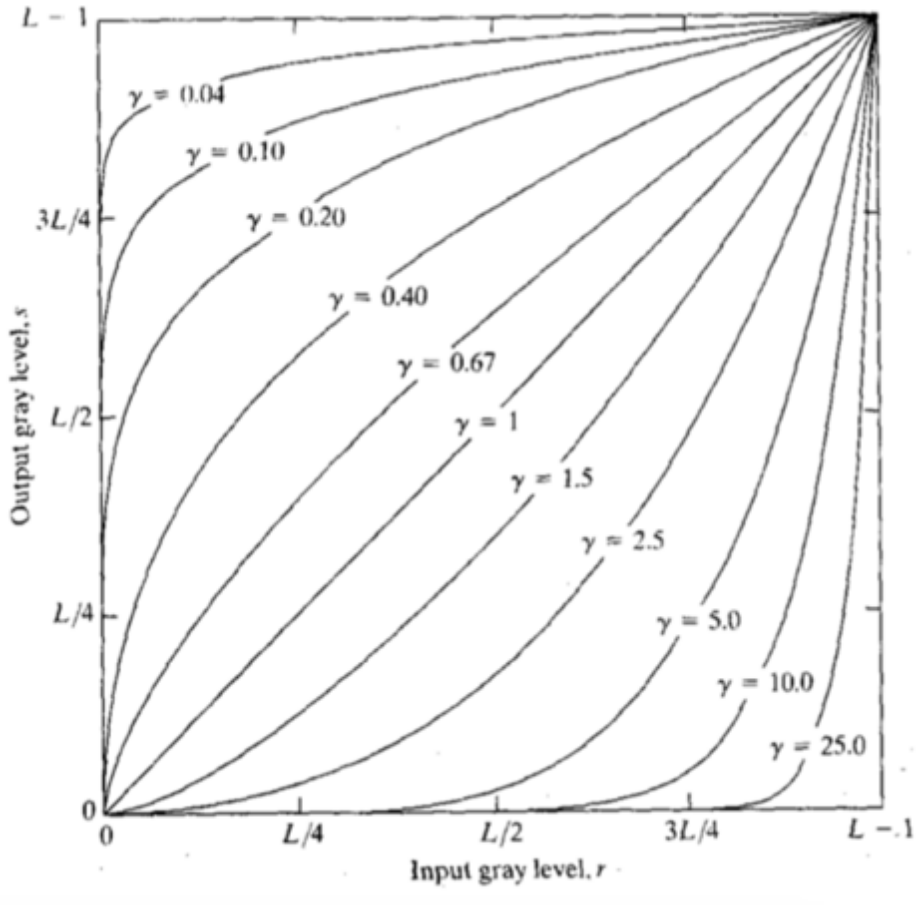

Figure 2.4: Plots of the gamma transformation grayscale conversion formula with various values of gamma (c = 1 in all cases).(Gonzalez and Wintz, 2010)

2.2 Image enhancement in spatial space

In image processing, the so-called spatial space refers to the image itself, and the operational object is pixels of grayscale image. Comparing to frequency space processing, spatial space processing is more computationally efficient and the processing results are easier to understand. Many spatial space processing techniques have been developed to meet a variety of different needs. Three techniques are introduced according to needs of this thesis.

2.2.1 Gray level transformation

The general form of grayscale transformation can be expressed as:

$$s = T(r) \tag{2.3}$$

in which s and r represent the grayscale of the pixel at the (x, y) position in the processed image and the original image, respectively, while T is the grayscale conversion formula. The performance of transformation depends on the choice of T. Figure 2.3 shows some basic gray-level transformation functions.

According to Figure 2.3, Logarithmic transformation can compress high grayscale and extend low grayscale to emphasize low grayscale areas of the image. Hence it can be used to enhance the underwater image in low light condition. The grayscale conversion formula is:

$$s = clog(1 + r) \tag{2.4}$$

One typical nth root grayscale transformation is called gamma transformation, with conversion formula as follow:

$$s = cr^{\gamma} \tag{2.5}$$

Gamma transformation can correct over-black image, which means it can also be utilized for underwater image enhancement. The performance of Gamma transform is related to the gamma value, as shown in Figure 2.4.

2.2.2 Histogram Equalization

Histogram is the grayscale statistical result of an image, and it is also a powerful tool in image enhancement. Histogram equalization is used to extend the dynamic range of an image, which can increase the contrast. Increasing the image contrast can improve the result of feature extraction. The simple histogram equalization can be described as:

$$s = (1 + (m/r)^{E})^{-1} \tag{2.6}$$

However, for underwater images, the contrast of different regions may vary greatly due to lighting conditions. Using simple histogram equalization cannot obtain good result. One alternative is the adaptive histogram equalization algorithm (AHE) . AHE based on the idea of block processing can handle this, but sometimes it will amplify unwanted noise.(Wikipedia contributors, 2018a) Another alternative is Contrast Limited AHE (CLAHE). A contrast threshold is introduced to remove the effects of noise. Detailed introduction and description of CLAHE can be found in (Zuiderveld, 1994).

2.2.3 Gray level slicing

When need to highlight or extract a region of an image, gray level slicing can be used. The gray level slicing will assign a higher grayscale to the region of interest.

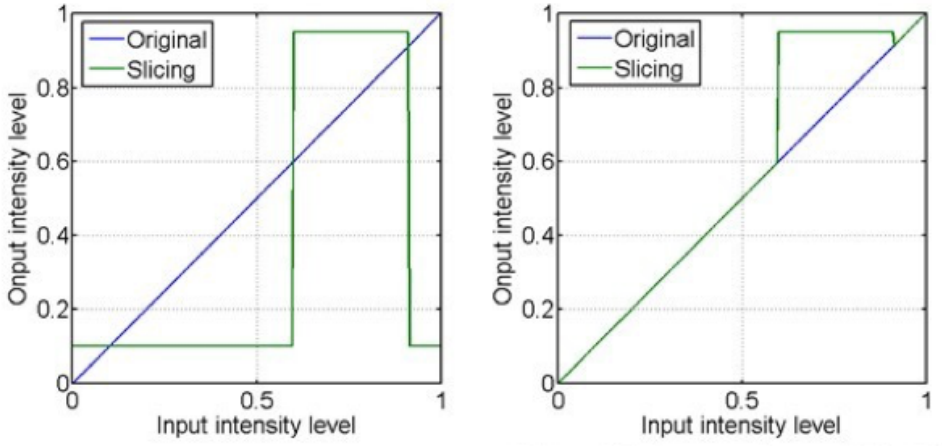

Figure 2.5: Gray level slicing.(Gonzalez and Wintz, 2010)

Gray level slicing has two types as shown in Figure 2.5. One type assigns a higher gray value to the region of interest while a lower gray value, usually zero, to the rest. This type will result in a binary image which contain only the region of interest. Another type just assigns a higher gray value to the region of interest while keep the rest unchanged. This type will result in a grayscale image with highlighted region of interest.

2.3 Morphology image processing

As binary images contain region of interest, they are usually inputs to algorithm such as object detection. Morphology processing is used on binary image before input to algorithm to improve the final result. Image morphology processing is based on mathematical morphology theory. Detailed mathematical derivation can be found in Chapter 9 of (Gonzalez and Wintz, 2010). In this chapter, we mainly introduce the processing effects of several image morphology processing methods.

Three basic morphological processing methods are introduced, including erosion, dilation and closing. Usually we need to define the foreground and background in a binary image before using morphological processing. But this is just a processing habit, which have no influence on morphological algorithm itself. Hence, we use color block instead of foreground and background.

2.3.1 Erosion

Erosion can reduce the area of white block, the corrosion effect depends on the area ratio of color block and structural element. For ratio larger than 1, the white block area is reduced, reduction scale depends on the shape of the structural element. However, for ratio smaller than 1, the white block area will disappear. Based on the effect of erosion, it can be used to remove noise or remove connected areas between white blocks. However, when the noise is removed, the area of the rest white blocks in the image is also affected. If we are concerned with the number or position of targets instead of the area, this impact is not significant.

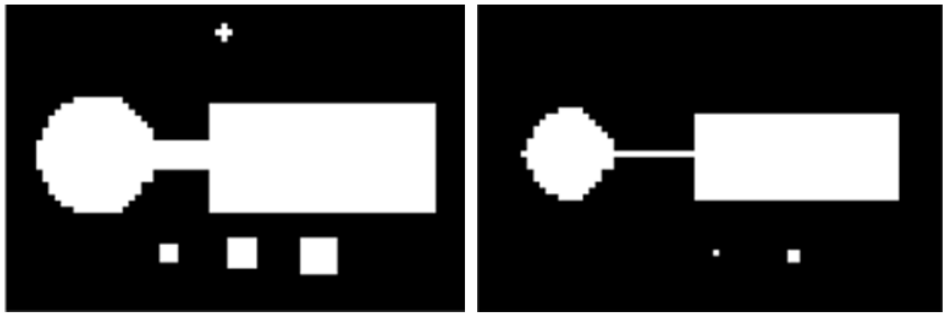

Figure 2.6: (a) Original image, (b) Erosion result.

2.3.2 Dilation

Dilation can expand the boundary of white block, the expand degree depends on the size of structural element. Dilation can enlarge the area of white block. Besides it is often used to bridge components belong to the same object which was broken after converting into binary image.

2.3.3 Closing

Performing erosion and dilation in a certain order can produce new processing effect, the closing is to first dilate and then erode. The closing operation can smooth the contour of white blocks, bridge narrow gaps and fill small holes.

Figure 2.7: (a) Original image, (b) Dilation result.

2.4 Image Segmentation

Image segmentation can be achieved based grayscale discontinuities or similarities. The method described in Section 2.2.3 is based on gray-scale similarity, that is, the image is divided into different regions according to a predetermined gray value criterion. The segmentation method introduced in this chapter is based on the discontinuity of grayscale, which is usually described by grayscale gradients.

2.4.1 Canny edge detector

There are many edge detection methods, but detailed descriptions and comparisons are not included in this thesis. The Canny edge detector is chosen in this thesis, due to the mature algorithm provided in OpenCV library and its better performance.

The Canny edge detector is a multi-step process. A standard Canny edge detector involves three steps:

1. Removing noise in the original grayscale image by Gaussian blurring.
2. Computing the Sobel gradient images both in x and y directions.
3. Determining edge based on a threshold value

To perform the Canny edge detector, the function 2.7 can be used:

$$cv2.Canny(src, c1, c2) \tag{2.7}$$

The first argument, src, is the blurred, grayscale image. Then, c1 and c2 represent two threshold value used for determining edges. Gradient value larger than c2 is considered to be an edge. And gradient value smaller than c1 is considered not to be an edge. As for

the gradient values locate between c1 and c2 are classified as edges or non-edges based on their intensities. Figure 2.8 shows the results of edge detection using different thresholds. Results show that the larger the selected thresholds, the less lines detected, but when the threshold is too small, the detected lines are not smooth. Therefore, the thresholds used in this thesis are: c1 = 3 and c2 = 40.

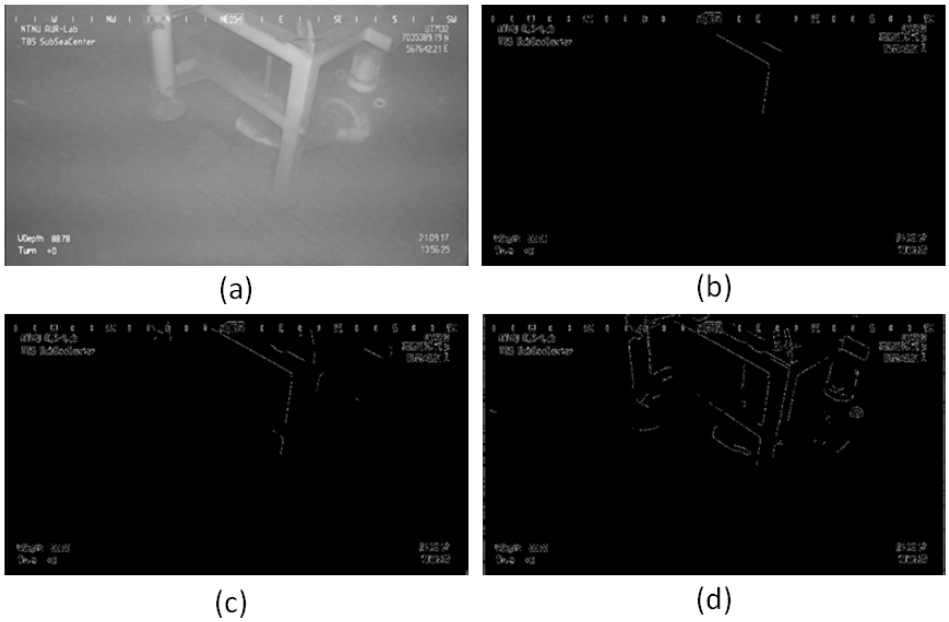

Figure 2.8: Canny edge detector results with different threshold values. (a) Original grayscale image (b) Canny result with c1 = 3 and c2 = 60 (c) Canny result c1 = 3 with c2 = 40 (d) Canny result c1 = 2 with c2 = 20

2.4.2 Hough Transformation

The result of Canny edge detector is a binary image which contains detected edges. Humans can obtain geometric information based on this binary image immediately and directly. But computers cannot obtain any geometric information directly from this binary image. This difference is mainly because people have a higher level of understanding of images than computers. So in order to allow the computer to obtain the same geometric information and to use it for pose estimation and guidance, we need to extract the line features further.

To use Hough transformation, the input image which is represented in Cartesian coordinate system need to be converted into polar coordinate system. This will result in point and line in Cartesian coordinate system become line and point in polar coordinate system as shown in Figure 2.9.

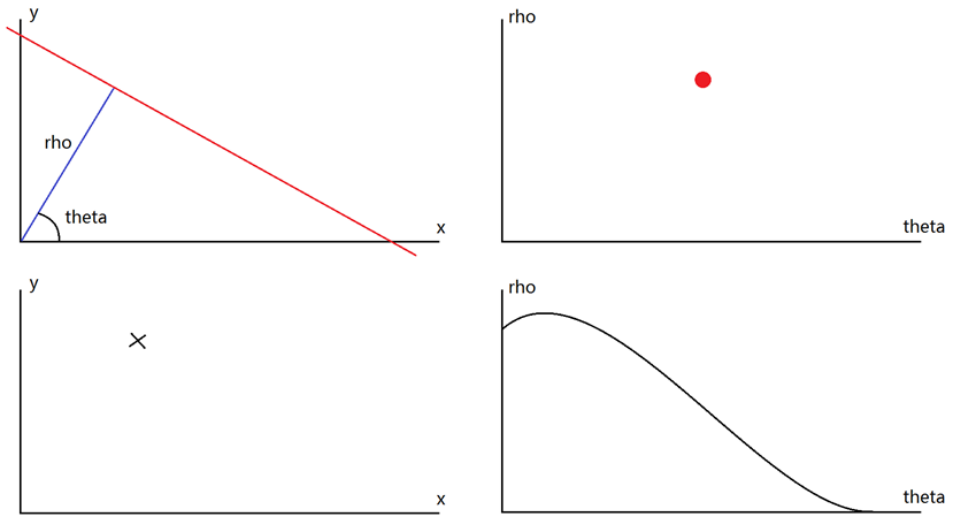

Figure 2.9: Dual relation between Cartesian cordinate and polar coordinate.Cartesian coordinate (left) polar coordinate (right).

Then detecting a straight line means finding the point where the curves in the corresponding parameter plane intersects the most, as shown in Figure 2.10.

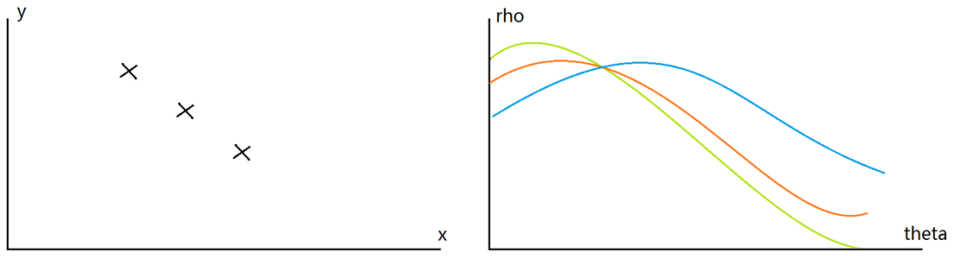

Figure 2.10: The three points on a straight line in the Cartesian coordinate system correspond to the three curves in the polar coordinate which intersect at one point.

Accumulating votes for these intersections, and then take out the points where the number of votes is greater than the pre-set minimum number of votes, that is the straight line detected in the original coordinate system.

After the Hough transform, we can obtain a list of points represented by (ρ, θ) in the polar coordinate system. Then the straight lines can be described by equation (2.8), in which x and y are coordinates value in Cartesian coordinates.

$$\rho = xcos(\theta) + ysin(\theta) \tag{2.8}$$

In this thesis, the $cv2.HoughLines$ functions provided by the OpenCV library is used directly for the algorithm development. It should be noted that the standard Hough transform does not consider the endpoints of line segments in the original image. Therefore, calculation of endpoints is needed to drawing the straight line in the original image. Since the endpoint information is not used in this algorithm, it is only mentioned here and no detailed calculation explanation is performed. Figure 2.11 shows the straight line detected by the Hough transform.

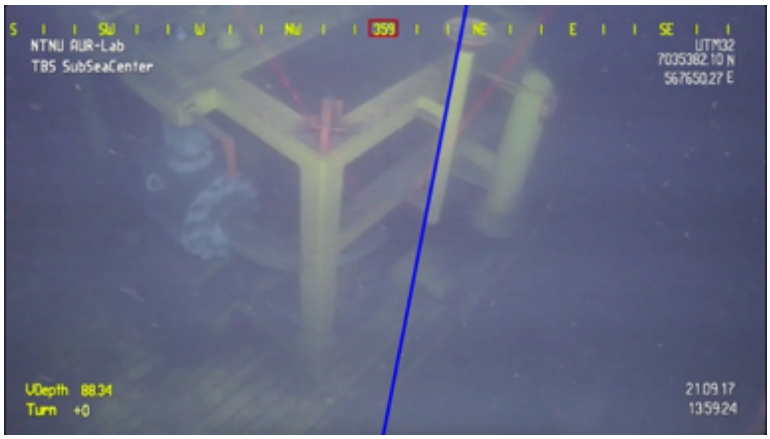

Figure 2.11: Canny edge detection result.

2.5 Epipolar geometry

The idea of visual guidance methods used in (Palomeras et al., 2016) and (Palmer et al., 2009) can be summarized as follow:

1. Extracting feature descriptor in the image

2. Calculating translation vector T and rotation matrix R based on two feature descriptors. It should be noted that one feature descriptor is obtained from the current camera image, and the source of another feature descriptor depends on the purpose of use. Feature descriptor extracted from previous frame image can be used for robot motion estimation. But feature descriptor obtained from destination image template can be used for calculating the desired motion for robot to get to the destination.

The feature descriptor and mathematical calculation method are two keypoints. The feature descriptors chosen in this thesis is introduced in section 2.6.1. The mathematical calculation method is introduced here.The mathematical calculation methods vary by image acquisition device:

1. When only the pixel coordinates of 2D points are known, for example by using monocular camera. Epipolar geometry can be used to calculate motion.

2. When the distance information is known, for example by using binocular camera or RGB-D, this problem can be solved by Iterative Closest Points (ICP).

3. When we have both the 3D points coordinates in space and the projection position in the camera, camera motion can be estimated through PnP.

Epipolar geometry is briefly introduced in this section, detailed mathematical derivation can be found in (Hartley and Zisserman, 2003), in which explanations of ICP and PnP are also given.

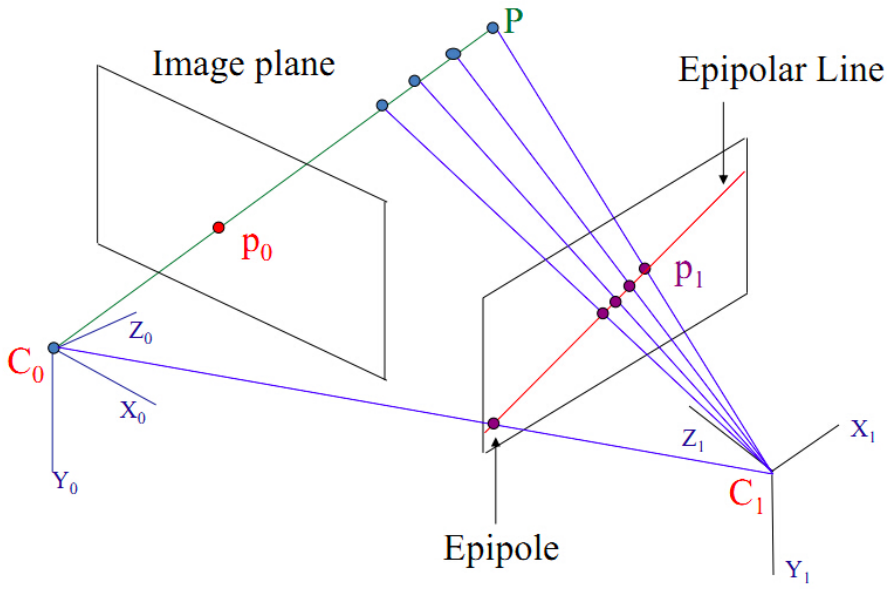

Figure 2.12: Epipolar geometry.

Assuming that features in two images are matched, we first consider one pair of the matching feature points, as shown in Figure 2.12. It is obviously that C_0, C_1 and P are coplanar, which can be described by vector expressions (2.9):

$$\overrightarrow{C_0p_0} \cdot (\overrightarrow{C_0C_1} \times \overrightarrow{C_1p_1}) = 0 \tag{2.9}$$

Then we can rewrite the vector expression obtained from coplanar relation as (2.10):

$$\vec{p_0} \cdot (t \times R\vec{p_1}) = 0 \tag{2.10}$$

in which, t represents the translation vector of the camera and R represents the rotation vector of the camera. It is known that the cross product between vector a and vector b can be represented by the dot product of a skew-symmetric matrix and vector b. The skew-symmetric matrix of vector a can be expressed as:

$$[a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.11}$$

Hence equation (2.10) can be rewrite as (2.12):

$$p_0{}^T [t]_\times R p_1 = 0 \tag{2.12}$$

in which the eigenmatrix was defined as $E = [t]_\times R$, and the equation 2.12 can be expressed as 2.13. The properties of essential matrix E can be found in (Hartley and Zisserman, 2003).

$$\begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0 \tag{2.13}$$

There are 8 unknown parameters in eigenmatrix E, in which E33 can be treated as scaling factor, hence we need at least 8 pairs of matched features. Once sufficient matched keypoints are obtained, we can utilize Singular Value Decomposition (SVD) to solve T and R which is explained detailed in (Ma et al., 2004).

As mentioned before, if we predefined a feature descriptor, such as a image of the destination used in SLAM method or a geometry feature descriptor used in Model-based method. Then use it to match the feature descriptors extracted from an image, and then calculate the motion of robot. The result can be used for robot navigation. Hence a successful feature matching is important.

2.6 Visual Guidance

2.6.1 Feature descriptor

Based on Sections 1.3 and 2.2, we need to choose the appropriate feature descriptor for the guidance algorithm in this thesis. Among the basic geometric features, we chose to use line features. The reason is:

1.The line feature extraction technology is mature and the extraction effect is good. Line features have been used in visual guidance in several scenarios, such as on-land self-driving cars tracking the road, underwater pipeline tracking and aircraft visually landing at the airport.

2.It is difficult to extract and utilize corner features for the scenarios discussed in this thesis. And the extraction technique of the region of interest is still immature.

2.6.2 Guidance strategy

For the three application scenarios of automatic driving, pipeline tracking and aircraft landing, the purpose of guidance is path tracking. As shown in the Figure 2.13, the detected

line features represent the path to be tracked, so the parallel relationship between line of sight and line features is the main basis for visual guidance.

Figure 2.13: Aircraft visual landing.

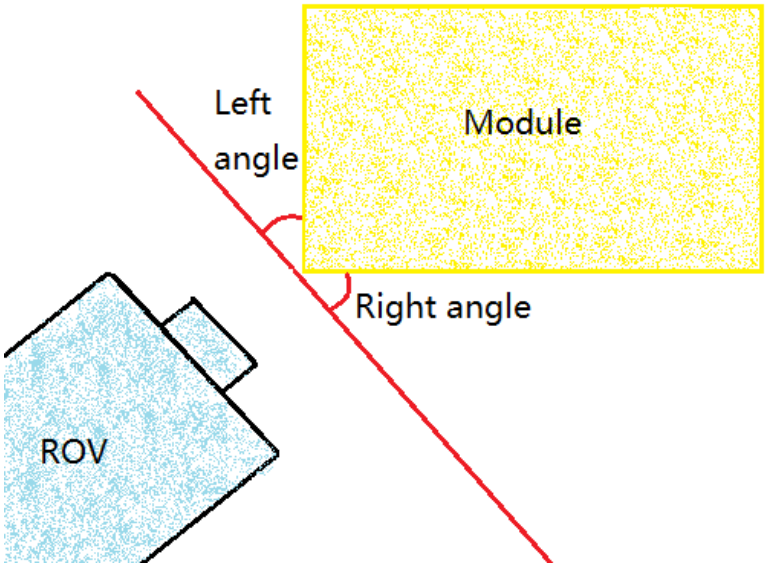

Figure 2.14: Camera image of the subsea module.

This thesis is not interested in path tracking of the ROV, but rather wants to move the ROV to the desired location for intervention. By observing the Figure 2.14 we can find that the angular relationship between the line of sight and the detected line features is the main basis for visual guidance in this thesis.

It should be mentioned that this thesis concerns about the valve intervention, and valves are located on the top surface of the subsea module according to the prior knowledge. Hence, visual guidance methods for vertical surface of the 3D subsea module is not discussed in this thesis. To ensure the ROV above the top, the altitude is set to be 3.5m which is measured by DVL. This not only simplifies the problem, but also ensures that the bottom of the ROV does not collide with the top of module.

The upper surface of the module that this article focuses on can extract two sets of line features, which are perpendicular to each other. Due to this, the visual guidance strategy proposed in this thesis is consist of two parts:

1.Rotating the ROV, let the line-of-sight perpendicular to one side of the upper surface.

2.Moving the ROV horizontally to the midpoint of this side, the midpoint is represented by the vanishing point of the line features.

By doing so, we can clearly define the search path for the valve area because we consider the direction and distance separately when operating the ROV, while SLAM and the model-based approach consider both.

2.6.3 Linear Perspective Theory

Based on a linear perspective projection method, real-world objects can be projected onto the camera plane. After the module is projected onto the image plane, the geometric features and relationships are still preserved, so we can extract the line features through the image and utilize them for guidance. In perspective projection, parallel lines parallel to the image plane remain parallel after being projected onto the image plane. Parallel lines that are at an angle to the camera plane intersect at a point in the image plane, which is called vanishing point.

For the cube module that this article focuses on, it can have three kinds of projections according to the number of vanishing points the image has:

1. One-point perspective as shown in Figure 2.15. This happens when the camera plane is parallel to one surface of the module. But this will not occur in this article. Because the camera on the ROV is installed at a 45 degree downward, which cannot parallel to any surface of the module.

2. Two-point perspective as shown in Figure 2.16. This happens when one set of lines is parallel to the camera plane. For this thesis, it happens when ROV is facing one side of module upper surface, which is the desired position for the first step of visual guidance strategy. In this case, when the vanishing point is on the vertical center line of the image plane, it indicates that the camera coincides with the symmetry axis of the structure. This is required in the second step of guidance.

3. Three-point perspective is the general imaging when ROV approaches the module in any orientation as shown in Figure 2.17. It is also the initial image condition of the visual guidance. In this thesis, we assume no roll and pitch angle when ROV observing the module which can simplify the lines classification as shown in Figure 2.18.

By looking downwards, the 3 types of lines included in the module structure will be distributed according to the Y shape as the white lines shown in Figure 2.18. The upper two branches represent lines on the upper surface while the lower vertical line is the vertical structural line.

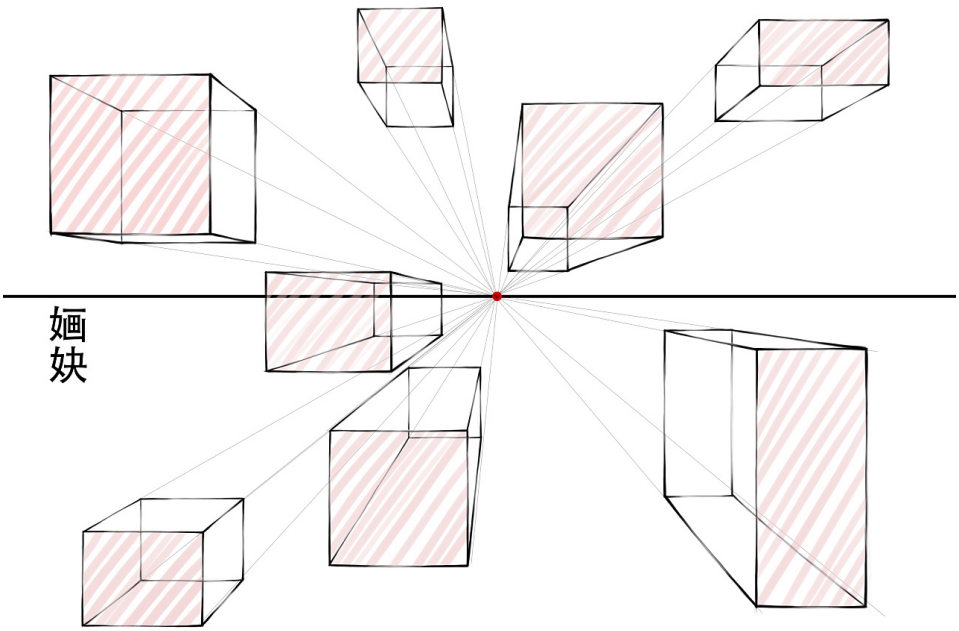

Figure 2.15: One-point perspective.

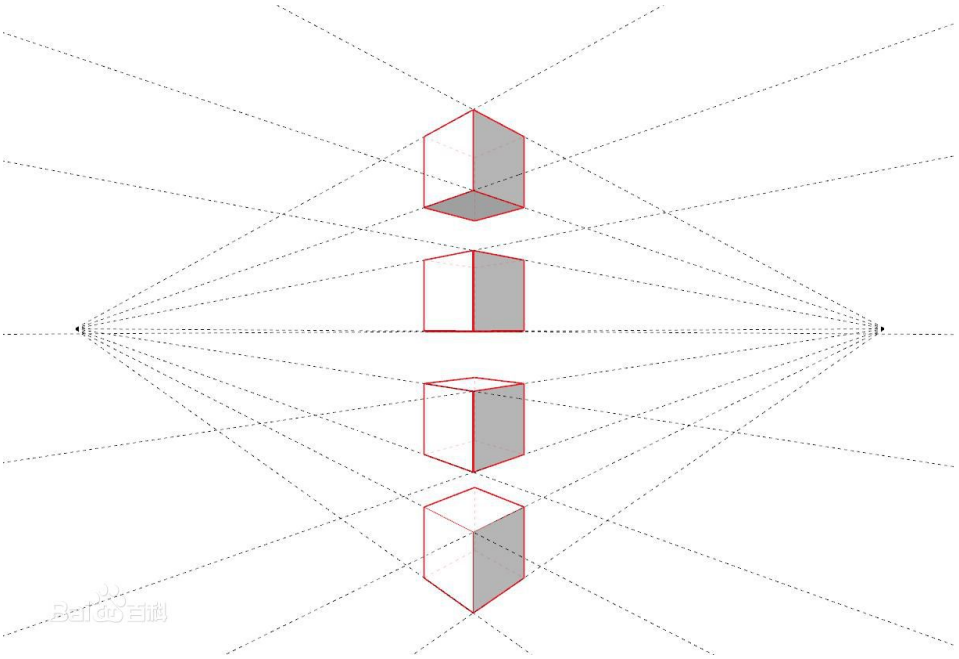

Figure 2.16: Two-point perspective.

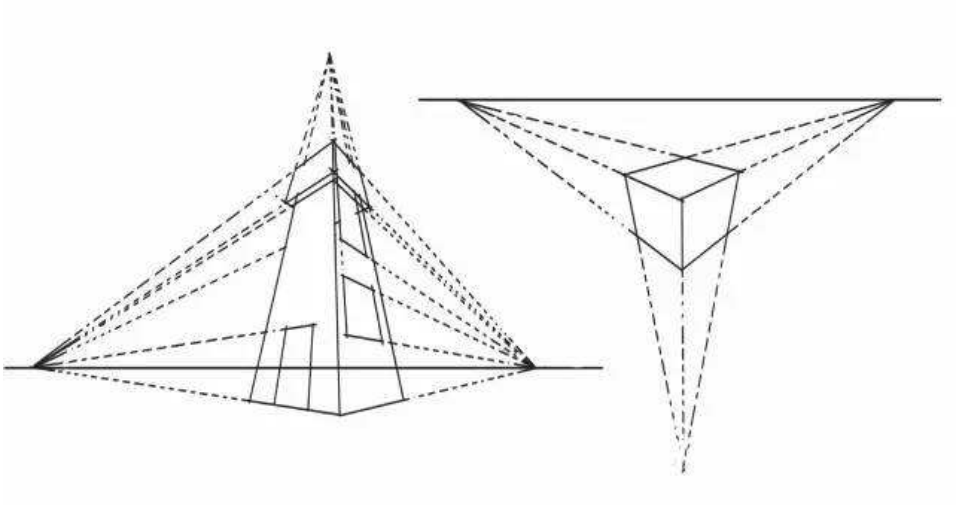

Figure 2.17: Three-point perspective.

Figure 2.18: Illustration of ”Y” type line classification.

2.6.4 Vision-system based motion decision making

Typically, ROVs are manipulated by human pilots based on camera feedback. The operational commands are based on human understanding of the underwater image content. Below we illustrate the decision making based on vision-system and summarize the principle in order to design the visual guidance algorithm.

For the situation shown in the Figure 2.19, assuming that we want to adjust the pose of the ROV so that it faces one side of the module upper surface. The general idea is that we first determine lines belonging to the upper surface from image and select one of them as the reference line. Then observing whether the reference line is parallel to the horizontal direction of the image when the ROV pose is adjusted. If it is parallel, it indicates that the ROV has reached the required posture. And in this case, the line-of-sight is perpendicular to the reference line, the angle between them is $90°$.

Figure 2.19: Case A.

For the situation shown in the Figure 2.20, assuming that we want to move the ROV horizontally so that it is at the centerline of the module upper surface. Generally, we can find the centerline based on image content. But in a general case, when the image lacks of artificial landmarks as shown in Figure 2.21, we can estimate the intersection of the vertical lines in the image plane, that is, the vanishing point. When the vanishing point is on the center line when we adjust ROV position, it means that the ROV is in the centerline of the structure.

Figure 2.20: Case B.

Figure 2.21: Image of corridor.

2.6.5 ROV pose adjustment

In Section 2.6.2, a visual guidance strategy was proposed. But the principles are described in human understanding manner, which is impossible for computer to achieve autonomous guidance based on these. For example, a human can determine whether a line is perpendicular to another line based on image content, but the computer needs a parametric description like angle relation. Therefore, the visual navigation strategy proposed in Section 2.6.2 is converted into the following description:

Step 1: Rotation of the ROV

1. From the point of departure, extract line feature in the current camera image, classify the lines detected and find the reference line, determine the rotation direction (left or right) and angle $r°$.

2. Move the ROV $85cm$ (which is half width of the ROV) horizontally to the (left or right), and then rotate the ROV an angle $\frac{r}{2}°$.

3. Repeat from step one until the rotation angle is close to $0°$.

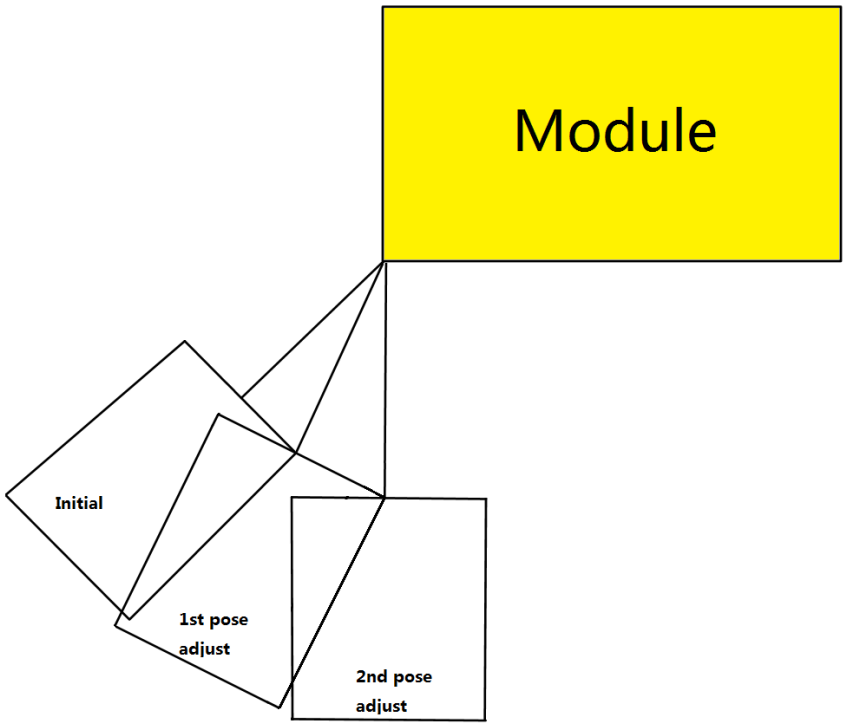

Figure 2.22: The ROV rotation path.

The Figure 2.22 shows the trajectory of ROV manipulating in this way, which looks like a N polygon. This result is consistent with the assumption of pi algorithm (Wikipedia contributors, 2018b), that is the circle can be treat as a N polygon. In the absence of distance information between ROV and module, we can use this method to move ROV according to an approximate circular trajectory.

Step 2: Movement of the ROV

1. Find the vanishing point of the current camera image.

2. Determine the movement direction of ROV based on relation between vanishing point and image centerline.

2.6.6 Line Classification

In order to achieve the visual guidance in accordance with the strategy proposed in Section 2.6.5, line features extracted by Hough Transformation need to be further processed. By analyzing the possible occurrences scenarios of the module structure lines, we find that it can be classified by $cos(\theta)$ and $sin(\theta)$. The parameterization criterion is shown in following Figures. In which blue lines are the lines extracted by Hough Transformation, while red lines are reference lines used to calculate (ρ, θ). The sine and cosine range of the corresponding theta angle are drawn on the right, this is the main basis for classifying line features.

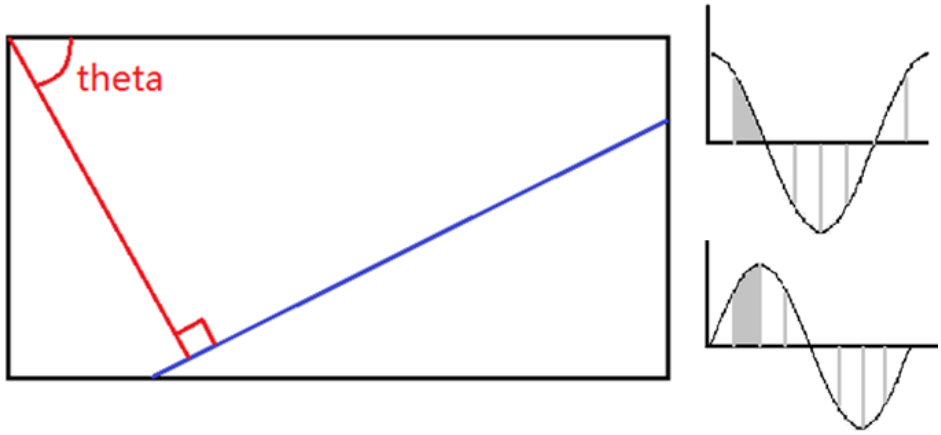

Figure 2.23: 1st type of lines in the camera image.

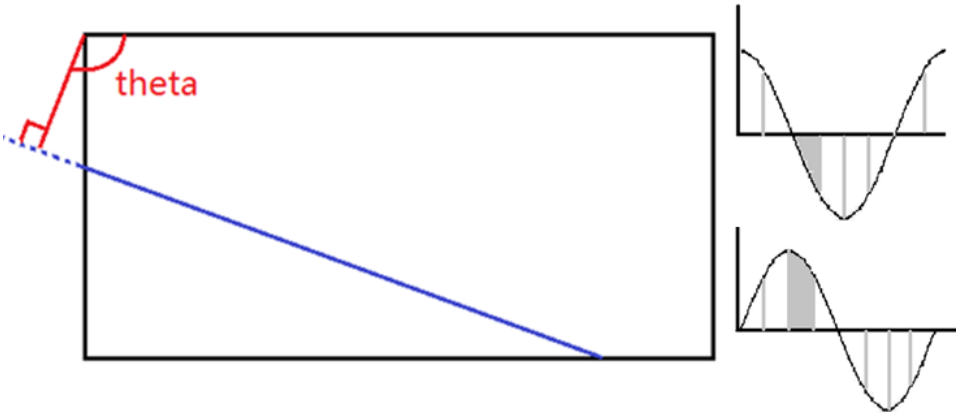

Figure 2.24: 2nd type of lines in the camera image.

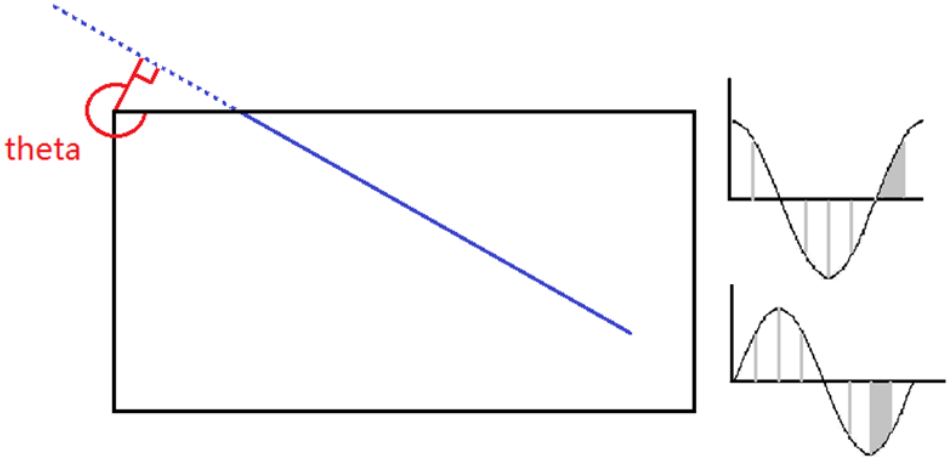

Figure 2.25: 3rd type of lines in the camera image.

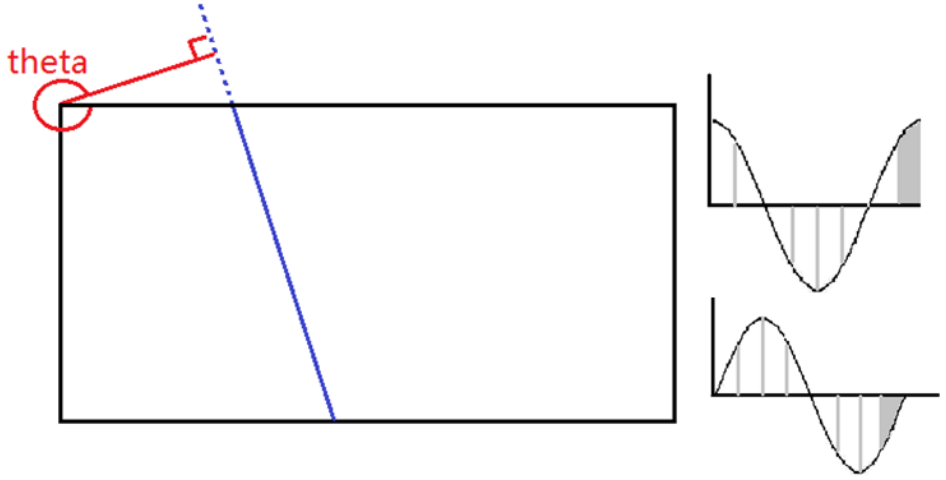

Figure 2.26: 4th type of lines in the camera image.

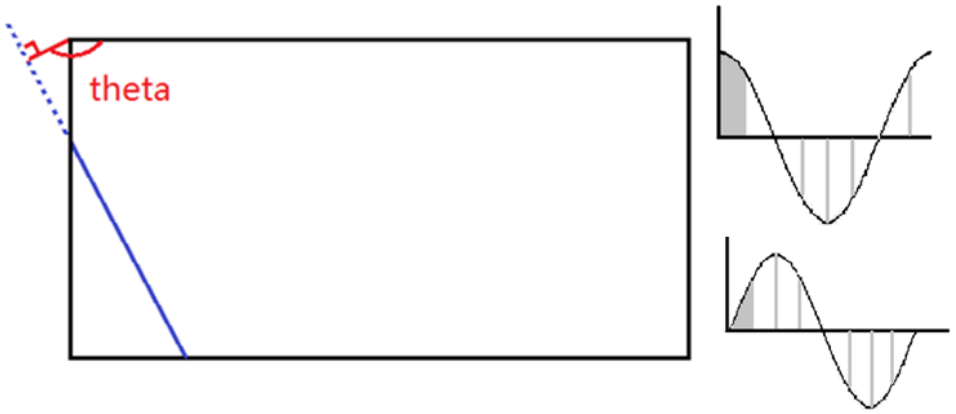

Figure 2.27: 5th type of lines in the camera image.

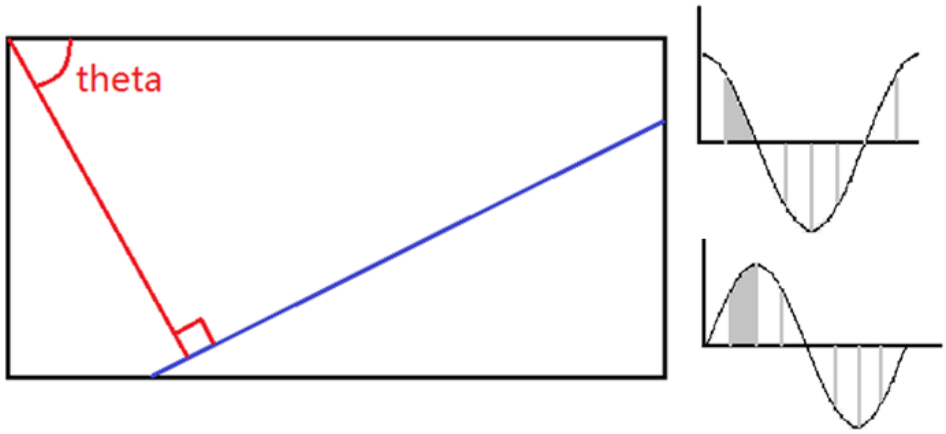

Figure 2.28: 6th type of lines in the camera image.

Chapter 3

Method

The image processing methods and visual guidance principles introduced in Chapter 2 are important tools for this thesis. This chapter describes how to combine these tools to achieve our goals. Two algorithm are implemented, Section 3.1 introduces close-range visual guidance algorithm based on line features, and Section 3.2 introduces a valve label detection algorithm based on shadow mode analysis. All the Python source code can be found in Appendix .

3.1 Close Range Visual Guidance

Utilizing the methods and theory introduced in Chapter 2, a close-range visual guidance algorithm is implemented.

3.1.1 Degree of freedom reduction

The ROV has 6 degrees of freedom, which are 3 translations and 3 rotations. According to the previous chapter, we can make the following assumptions to simplify the problem and reduce the degree of freedom from 6 to 3:

1. Let the ROV's altitude always be greater than the module height. The altitude value can be measured by DVL. The desired altitude is set to 3.5m. This is because the camera is installed at a 45° downwards, and this altitude value ensures that the upper surface of the module can be captured.

2. Let the roll and pitch angles be zero at all waypoints, which facilitates the classification of line features.

3.1.2 Scenarios

When the ROV approaches the module from a random orientation, from a top view, there are only two scenarios due to the structural symmetry of the module as shown in Figure 3.1. For the first scenario only one side of the upper surface can be detected as shown in Figure 3.2. But for the second scenario, both two sides of module upper surface can be detected as shown in Figure3.3.

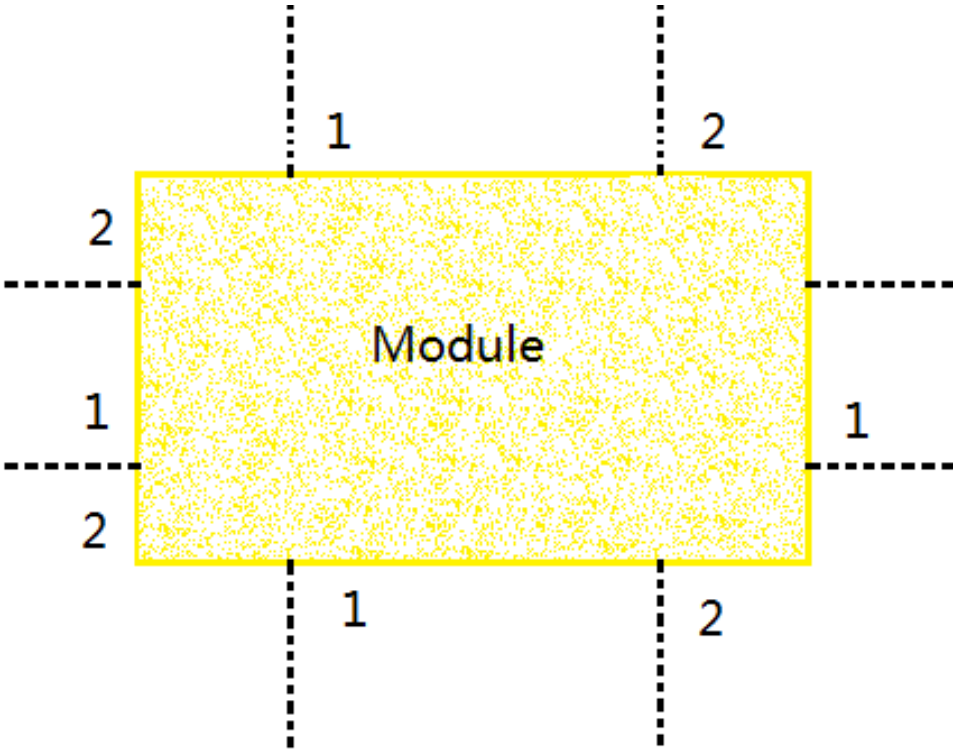

Figure 3.1: Division of area around the panel (top view).

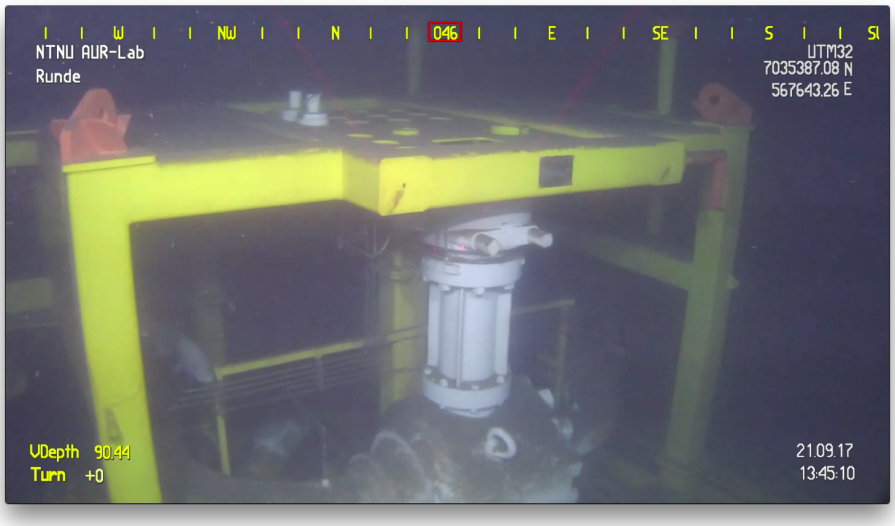

Figure 3.2: Scenario 1: only one side can be detected.

Figure 3.3: Scenario 2: both two sides can be detected.

3.1.3 Algorithm procedure

Explanation of each module in the close-range visual guidance is given in the following sections. The corresponding python source codes are included in Appendix.

Image Pre-processing

The purpose of this pre-processing is to reduce the repetitive operation of subsequent processing. Firstly, the original RGB image is converted to a grayscale image and a HSV image. And the HSV image is split into three channels. The size of the image is also calculated. Finally, these parameters are returned which can be called by other modules.

Removal of Interfere Information

The raw image obtained by camera is shown in Figure 3.4.

Figure 3.4: Raw image capture by the camera.

There is some text information around the image. These are useful but can interfere with image feature detection. Hence a mask is created to filter out the interfere information. The mask is based on histogram analysis on HSV color space. Figure 3.5 shows the mask created for Figure 3.4.
The mask is used for the output of Canny edge detector. Figure 3.6 shows the output of Canny edge detector. And Figure 3.7 shows the image using mask, in which all interfere information were disappeared.

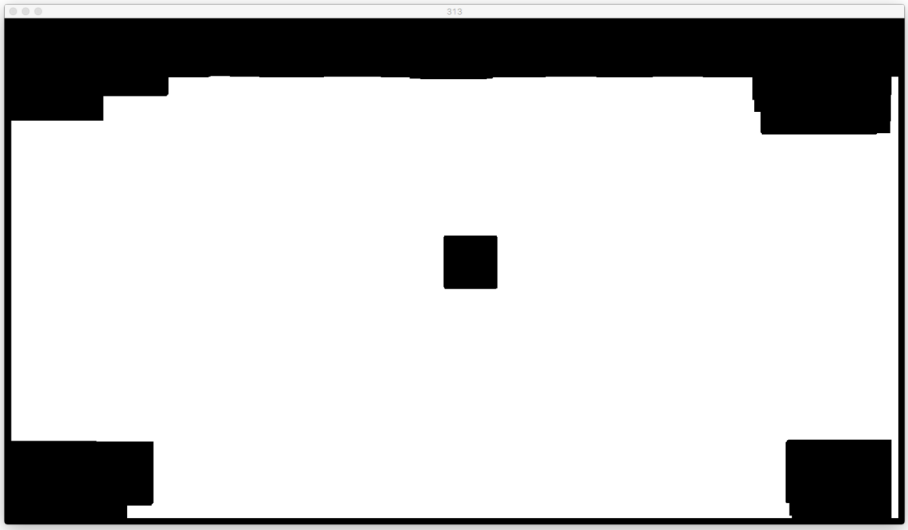

Figure 3.5: Mask created for raw image.

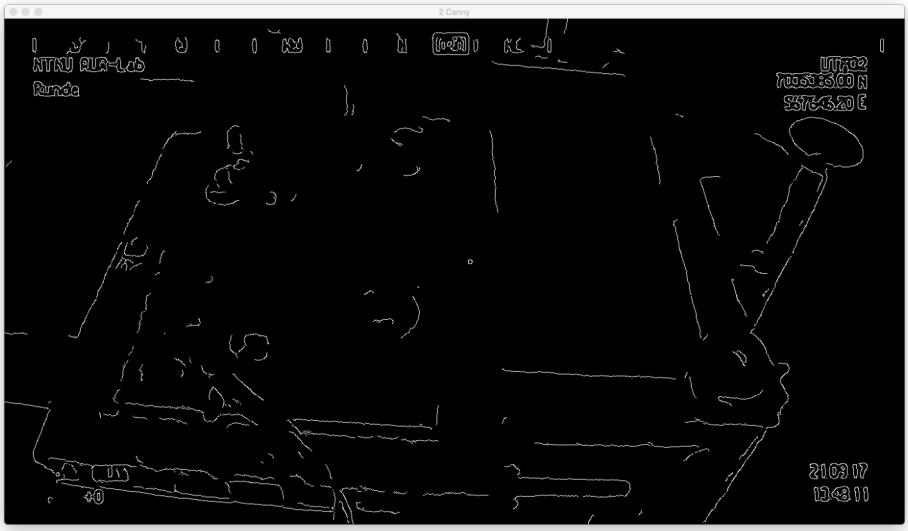

Figure 3.6: Output of Canny edge detector.

Figure 3.7: Canny edge detection result after using mask.

Image enhancement and lines extraction

The line feature is extracted by Canny edge detector and Hough transformation. But the image enhancement method is modified. The CLAHE performs good when enhance the camera image. But due to the extraction criterion of Hough Transformation, enhancement of non-structural line part will lead to incorrect extraction as shown in Figure 3.8 (CLAHE). Hence we need an image enhancement method which only improve the contrast around structural line.

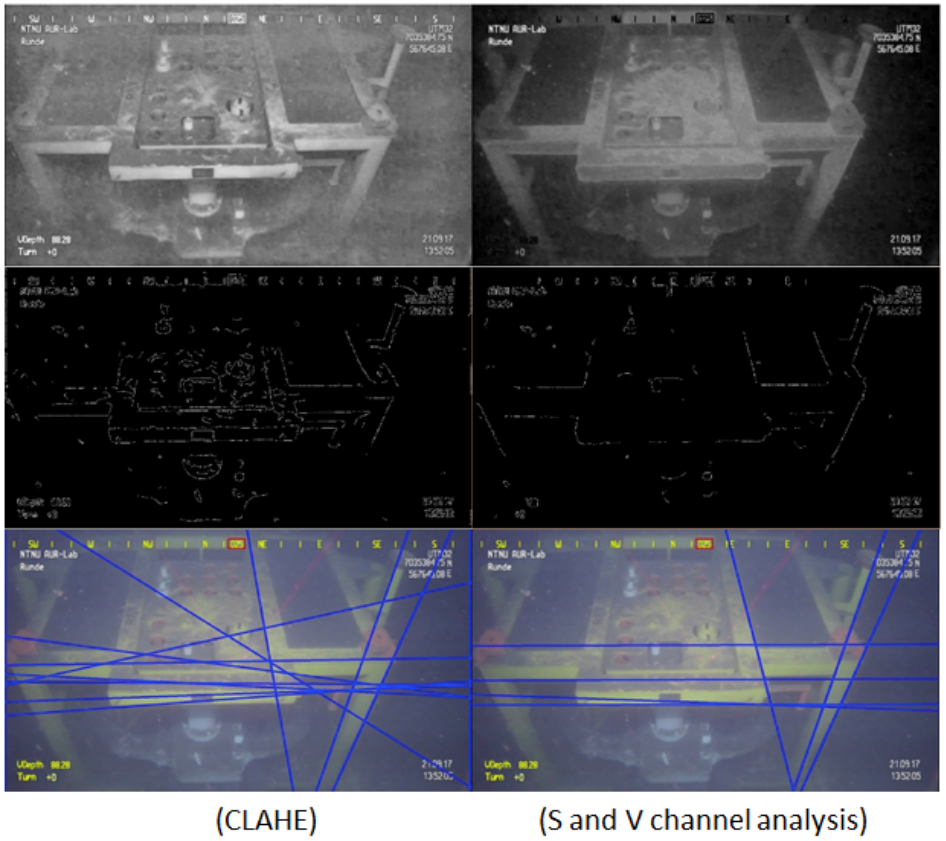

Figure 3.8: Hough Transformation result comparison: CLAHE (left), HSV(right)

Through histogram analysis on HSV color space, we found the Saturation channel and Value channel can be used to enhance the structural-line as shown in Figure 3.9 and Figure 3.10 . The grayscale of module region in Value channel is higher than Saturation channel.

Figure 3.9: Saturation channel of the camera image.

Figure 3.10: Value channel of the camera image.

By subtracting Saturation channel from Value channel, the contour of module will be enhanced as shown in Figure 3.8 (S and V channel annalysis). Figure 3.8 shows that this method can reduce the incorrect extraction.

Duplicate Line Removal

Using the Hough transform, we can detect a large number of eligible line features. But often there are multiple repeating lines at one edge of a structure as shown in Figure 3.11. So we need to remove the extra lines before classifying the remaining lines.

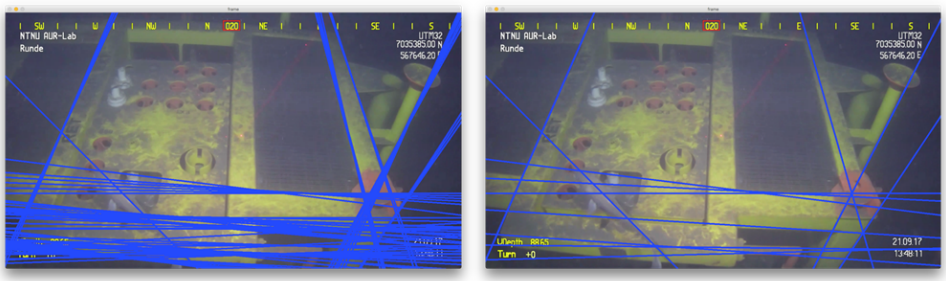

Figure 3.11: Removal of duplicate lines.

We loop through all the lines and calculate the difference in theta angle between the lines. If the difference is in the range of $(0, 0.25rad)$ or $(2.89, 3.39)$, these two lines are considered to be parallel lines. If the rho difference between parallel lines is less than 40 pixels, they are considered as duplicate lines and one of them will be deleted from the list.

After processing, the remaining lines can be classified according to the criteria introduced in section 2.3.6.

Line Classification

After all duplicate lines are removed from the line list, the classification can be carried out according to the classification criterion proposed in Section 2.6.6. The implementation can refer to the Python source code in Appendix.

vanishing points

Several vanishing point detection algorithms have been proposed such as J-linkage.(Toldo and Fusiello, 2008) The vanishing point play an important role in Computer Vision application .(Kweon, 2012) As for this thesis, the vanishing point is the important reference point for the ROV horizontal motion. However, this thesis does not use complex vanishing point detection algorithm, such as J-linkage, the coordinate of lines intersection is calculated to represent the vanishing point.

3.2 Label Detection

The purpose of close-range visual guidance is to move the ROV to the intervention region, where valves located. Once this goal is achieved, the next step for the autonomous intervention is to detect the intervention object. Valves on the module are the intervention object in this thesis.

3.2.1 Strategy of valve detection

As discussed in Introduction chapter, challenges faced in valve detection are large number of valves and the existence of labels and auxiliary structures. In the case that the structure cannot be simplified, we need to combine these to detect the valve. Therefore two strategies are proposed for valve detection:

1. Find out center point of valves first, then try to locate the corresponding labels to identify the valve.
2. Detect the label first, then recognize the content of labels. Once labels are identified, location of corresponding valves can be find easily based on prior knowledge of valves.

The first strategy cannot solve the problem. When the valve area is detected, only the spatial information of this area, i.e. the two-dimensional coordinates in the image plane, is obtained. Only relying on this information, can not determine the exact location of the corresponding label, and can not clearly know the function of the detected valve.

The second strategy can solve this problem. Two information can be obtained from the label recognition, that is the label two-dimensional coordinates in the image plane and content of the label which indicate the function of corresponding valve. Once the label is recognized, the corresponding valve can be located easily based on prior knowledge. Hence the second strategy is chosen for this thesis.

3.2.2 Algorithm procedure

Explanation of each module in the label detection algorithm is given in the following sections. The corresponding python source codes are included in Appendix.

Shadow model analysis

OCR has been applied in many areas, such as automatic number-plate recognition. There are two ways to implement OCR. The traditional one is to segment the character region first and then recognize it.(Wikipedia contributors, 2018c) The other is based on the machine learning method that does not need to be segmented and directly recognizes on the original image. For underwater images under poor illumination processed in this paper, it is not efficient to use machine learning method. Hence the traditional method is chosen for this thesis. And the segmentation part has been implemted in this thesis.

Both histogram analysis and grayscale slicing are typical methods used for segmentation. But they cannot give ideal segmentation result in underwater image due to the poor illu-

mination condition. Hence, the problem of uneven illumination needs to be solved first. By using the Top-hat operation, a shadow mode of the original grayscale image can be obtained. Examples and explanation of Top-hat can refer to (Gonzalez and Wintz, 2010), and Figure 3.12 shows the shadow mode of original image. This shadow mode is obtained by using a quite small structural element, which can remove the dark region of the original image.

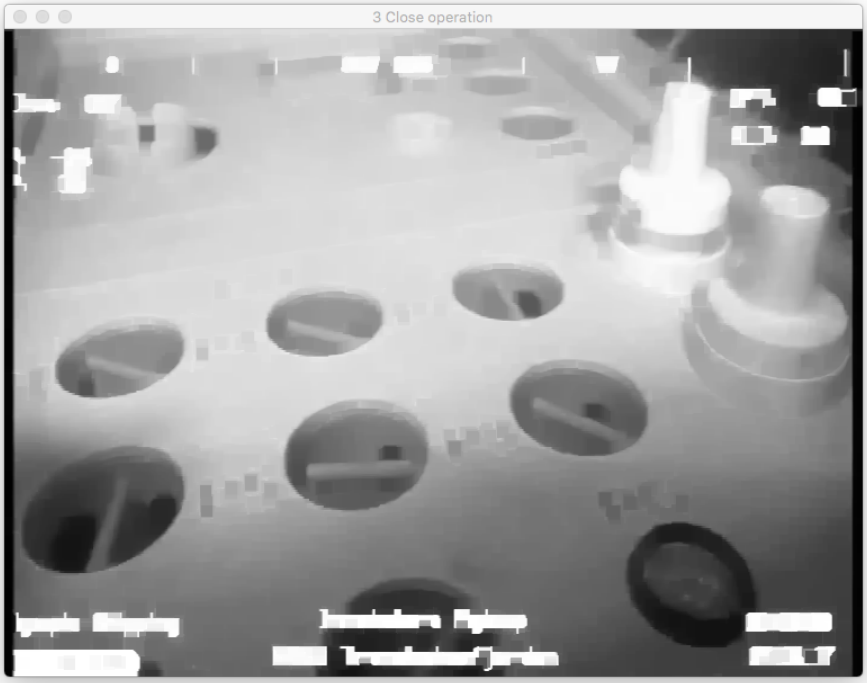

Figure 3.12: Shadow mode analysis of image.

Subtraction in OpenCV library

The subtract mention in OpenCV is different from the traditional definition in mathematical. When we processing subtract for image using the OpenCV library, it will perform clipping and ensure pixel values never fall outside the range $(0, 255)$. Due to this property, the label can be highlighted by subtract the original grayscale image from the shadow mode obtained from closing. Figure 3.13 shows the result.

Figure 3.13: Remove the shadow from grayscale image.

Histogram equalization

The result image of OPenCV subtraction is a dark image with low contrast. The simple histogram equalization is applied to enhance the overall contrast. The label is much more visible after processing as shown in Figure 3.14.

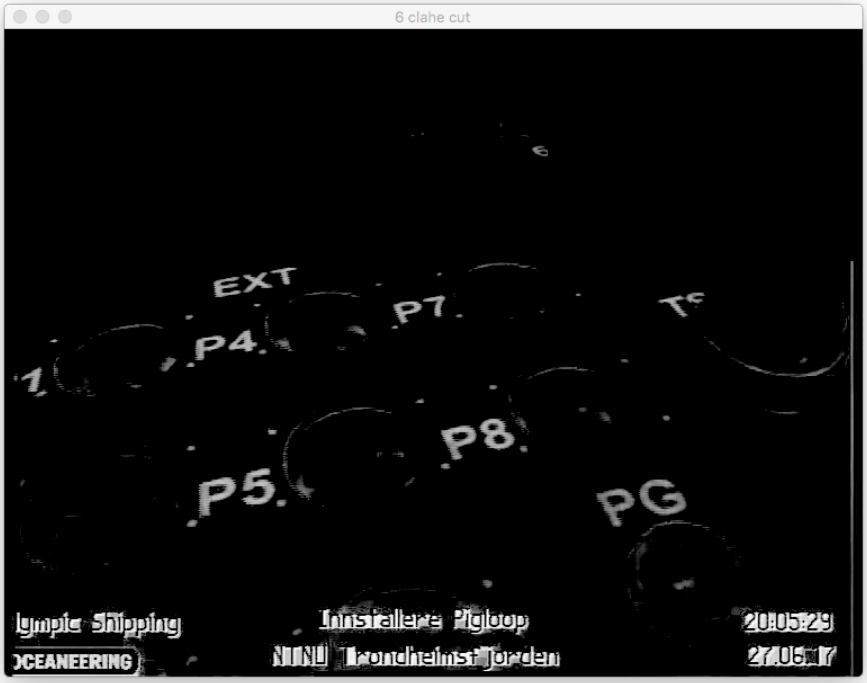

Figure 3.14: Histogram equalization.

Otsus method

Some structure edges are highlighted after histogram equalization. In order not to interfere with the segmentation of the labels, Otsu's method is used to remove these edges. This method is very suitable for solving this problem. Otsu's method assumes that the image contains two types of pixels after the bi-modal histogram (foreground pixels and background pixels)(Wikipedia contributors, 2018g), which is the same as the output image in Section 3.2.2.3. The optimal threshold is then calculated to separate the two classes. Figure 3.15 shows the results of the Otsu threshold.

Figure 3.15: Otsu's thresholding.

Find contour

The contour is the starting point of object segmentation. The contour can be extracted by calling the function in OpenCV library. Detail explanation can be found in OpenCV instruction ($https://docs.opencv.org$). To improve the correct detection rate, a filter based on label aspect ratio can be implemented. Details of contour filter can be found in the python source code. The Figure 3.16 shows the final label detection result, in which the label is enclosed by red rectangular.

Figure 3.16: Contours of detected labels.

Chapter 4

Result

The algorithm developed in this thesis is tested on two video, which are recorded during ROV intervention missions as shown in Figure 4.1 and 4.2. The results of video test are presented by capture of Keyframe.

Figure 4.1: Video used for testing close-range visual guidance algorithm.

Figure 4.2: Video used for testing label detection algorithm.

4.1 Close Range Visual Guidance result

There are two test targets for this algorithm:

1. Whether rotation angle and direction can be obtained from extracted line features. Whether the direction of the auxiliary arrow displayed on the Keyframe is consistent with the human judgments.

2. Whether the position of structure center line can be determined (represented by vanishing points). Whether the direction of the auxiliary arrow is consistent with the human judgments.

Rotation of the ROV

Figure 4.3 is the key frame captured when the ROV is rotating.

Figure 4.3: Rotation angle and direction

The yellow line represents one side of module upper surface. The white straight line is a line which parallel to image frame. Blue lines are all lines extracted by Hough Transformation. The white line is used as a reference to determine whether the ROV is face to one side of the module upper surface.

The rotation angle are present in green. It is 5.5 degree for this Keyframe. Humans cannot directly judge the exact angle value. Hence a green arrow was drawn to show the rotation direction that algorithm suggested which is consistent with us judgment.

We can conclude from Figure 4.3 that the rotation angle and direction can be obtained from extracted line features and the direction of the auxiliary arrow displayed on the Keyframe

is consistent with the human judgments.

Horizontal movement of the ROV

Figure 4.4 is the key frame captured when the ROV is moving horizontally.
The white point is the projection of the vanishing point on the image plane. The white line is the vertical centerline of the camera plane. The green arrow shows the horizontal movement direction that algorithm suggested which is consistent with us judgment.

We can conclude from Figure 4.4 that the position of structure center line can be determined (represented by vanishing points) and the direction of the auxiliary arrow is consistent with the human judgments.

The implemented close-range visual guidance algorithm is functioning properly.

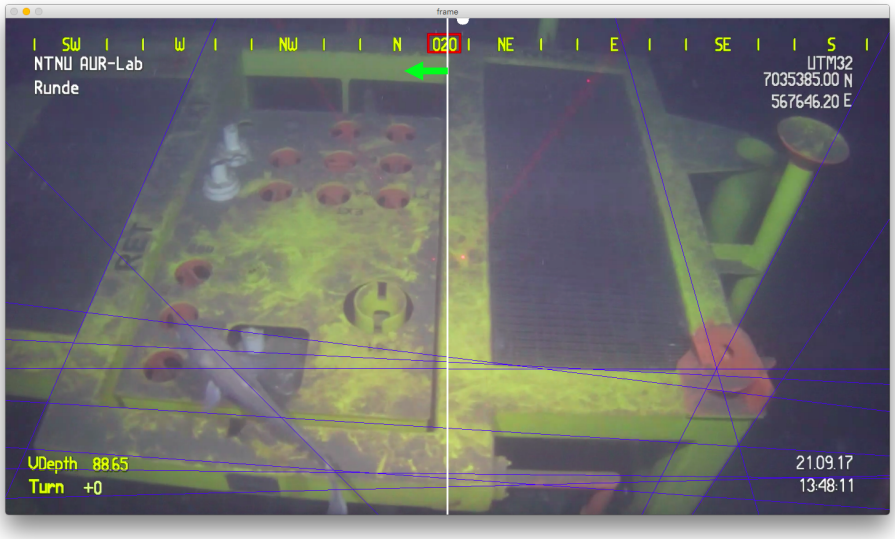

Figure 4.4: Horizontal movment direction

4.2 Label Detection

The test goal of this algorithm is to verify whether the label can be successfully detected.

4.2.1 Label detection result

The Figure 4.5 presents the label detection result in one Keyframe. Non-label area exist in the binary image are filtered out and the detected labels are enclosed in red rectangular.

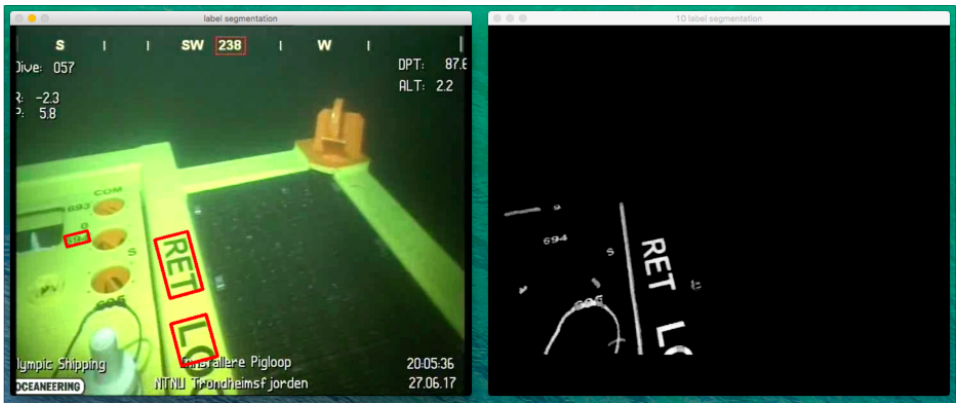

Figure 4.5: Label detection with contour filter

4.2.2 Bad illumination

The label detection algorithm developed based on shadow mode analysis, which make it robust to bad illumination such as low illumination and overexposure. The detection results of Overexposure and low illumination are difficult to capture in a video as these bad illumination condition not last for a long time. The python source code is provided in Appendix, reader can test the detection performance.

Chapter 5

Analysis and Discussion

5.1 Uncertainties and Obstacle

5.1.1 Field experiment

The close-range visual guidance algorithm was tested with video. Feature extraction and parameter calculation functions are all working fine. However, although the results show that the proposed motion trend is consistent with human judgment, whether the calculated motion parameters are correct requires field experiment to determine.

5.1.2 Dusty surface

In Section 1.3.1, we mentioned that ocean snow can cause dust on the upper surface of the module, causing image characteristics to change. This situation has an impact not only on navigation using the SLAM method, but also has an effect on label detection. Therefore, after ROV move to the valve area, the surface dusting is required before label detection.

5.2 Pros and cons of the methods

The advantage of the algorithm implemented in this thesis is that the computational complexity is small which make it can be run in real time and very robust. This is because of mathematics or geometric models used for developing algorithm are basic. The drawback is that visual navigation lacks depth information, and due to the approximate method used to estimate motion trajectory, the trajectory precision is low.

Chapter 6

Conclusion and future work

6.1 Conclusion

The results shown in Chapter 4 implies that all algorithms developed in this thesis are successful. The suggestion of the rotation direction offer by the algorithm is consistent with human judgment, which is same for the horizontal movement. And the label detection algorithm is robust.

6.2 Further work

6.2.1 Marine Snow

The presence of ocean snow causes precipitation of dust on the underwater module which change the feature on module surface. A solution is required to deal with this not avoidable problem. There are two possible solutions, one is to develop autonomous dust removal function, another is to use new type of sensors instead of the camera.

6.2.2 Label recognition

In this paper, only the detection and segmentation of the label is completed, and the identification work needs to be completed in the future. After the label recognition function is completed, posture of ROV can be adjusted based on label features.

Bibliography

Brusletto, L. S., 2016. Computer vision based obstacle avoidance for a remotely operated vehicle. Master's thesis, Norwegian University of Science and Technology.

Carrera, A., Palomeras, N., Hurts, N., Kormushev, P., Carreras, M., 2015. Cognitive system for autonomous underwater intervention. Pattern Recognition Letters 67, 91 – 99, cognitive Systems for Knowledge Discovery.
URL http://www.sciencedirect.com/science/article/pii/S0167865515001725

Choi, S. K., Takashige, G. Y., Yuh, J., Jul 1994. Experimental study on an underwater robotic vehicle: Odin. In: Autonomous Underwater Vehicle Technology, 1994. AUV '94., Proceedings of the 1994 Symposium on. pp. 79–84.

Evans, J., Redmond, P., Plakas, C., Hamilton, K., Lane, D., Sept 2003. Autonomous docking for intervention-auvs using sonar and video-based real-time 3d pose estimation. In: Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492). Vol. 4. pp. 2201–2210 Vol.4.

Evans, J. C., Keller, K. M., Smith, J. S., Marty, P., Rigaud, O. V., 2001. Docking techniques and evaluation trials of the swimmer auv: an autonomous deployment auv for work-class rovs. In: MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295). Vol. 1. pp. 520–528 vol.1.

Farr, N., Bowen, A., Ware, J., Pontbriand, C., Tivey, M., May 2010. An integrated, underwater optical /acoustic communications system. In: OCEANS 2010 IEEE - Sydney. pp. 1–6.

Fletcher, B., Young, C., Buescher, J., Whitcomb, L. L., Bowen, A., McCabe, R., Yoerger, D. R., Sept 2008. Proof of concept demonstration of the hybrid remotely operated vehicle (hrov) light fiber tether system. In: OCEANS 2008. pp. 1–6.

Gonzalez, R. C., Wintz, P., 2010. Digital image processing. Prentice Hall International 28 (4), 484 – 486.

Hartley, R., Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge University Press.

Kweon, I., 2012. Globally optimal line clustering and vanishing point estimation in manhattan world. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 638–645.

Lane, D. M., Davies, J. B. C., Casalino, G., Bartolini, G., Cannata, G., Veruggio, G., Canals, M., Smith, C., O'Brien, D. J., Pickett, M., Robinson, G., Jones, D., Scott, E., Ferrara, A., Angelleti, D., Coccoli, M., Bono, R., Virgili, P., Pallas, R., Gracia, E., Dec 1997. Amadeus: advanced manipulation for deep underwater sampling. IEEE Robotics Automation Magazine 4 (4), 34–45.

Ma, Y., Soatto, S., Ko?Eck, J., Sastry, S. S., 2004. An invitation to 3-D vision :. Springer,.

Marani, G., Choi, S. K., Yuh, J., 2009. Underwater autonomous manipulation for intervention missions auvs. Ocean Engineering 36 (1), 15 – 23, autonomous Underwater Vehicles.
URL http://www.sciencedirect.com/science/article/pii/S002980180800173X

Maurelli, F., Krupinski, S., Petillot, Y., Salvi, J., Sept 2008. A particle filter approach for auv localization. In: OCEANS 2008. pp. 1–7.

Nagappa, S., Palomeras, N., Lee, C. S., Gracias, N., Clark, D. E., Salvi, J., 2013. Single cluster phd slam: Application to autonomous underwater vehicles using stereo vision. In: OCEANS-Bergen, 2013 MTS/IEEE. IEEE, pp. 1–9.

Ozog, P., Eustice, R. M., May 2014. Toward long-term, automated ship hull inspection with visual slam, explicit surface optimization, and generic graph-sparsification. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 3832–3839.

Palmer, T., Ribas, D., Ridaoy, P., Malliosy, A., 2009. Vision based localization system for auv docking on subsea intervention panels. In: Oceans 2009-Europe. IEEE, pp. 1–10.

Palomeras, N., Carrera, A., Hurtós, N., Karras, G. C., Bechlioulis, C. P., Cashmore, M., Magazzeni, D., Long, D., Fox, M., Kyriakopoulos, K. J., et al., 2016. Toward persistent autonomous intervention in a subsea panel. Autonomous Robots 40 (7), 1279–1306.

Palomeras Rovira, N., Peñalver, A., Massot Campos, M., Negre, P. L., Fernández, J. J., Ridao Rodríguez, P., Sanz, P. J., Oliver Codina, G., 2016. I-auv docking and panel intervention at sea. Sensors, 2016, vol. 16, núm. 10, p. 1673.

Ribas, D., Ridao, P., Neira, J., 2010. Underwater SLAM for structured environments using an imaging sonar. Vol. 65. Springer.

Rigaud, V., Coste-Maniere, E., Aldon, M. J., Probert, P., Perrier, M., Rives, P., Simon, D., Lang, D., Kiener, J., Casal, A., Amar, J., Dauchez, P., Chantler, M., Mar 1998. Union: underwater intelligent operation and navigation. IEEE Robotics Automation Magazine 5 (1), 25–35.

Rist-Christensen, I., 2016. Autonomous robotic intervention using rov. Master's thesis, Norwegian University of Science and Technology.

Salvi, J., Petillo, Y., Thomas, S., Aulinas, J., 2008. Visual slam for underwater vehicles using video velocity log and natural landmarks. In: OCEANS 2008. IEEE, pp. 1–6.

Sanz, P. J., Ridao, P., Oliver, G., Casalino, G., Insaurralde, C., Silvestre, C., Melchiorri, C., Turetta, A., 2012. Trident: Recent improvements about autonomous underwater intervention missions. IFAC Proceedings Volumes 45 (5), 355 – 360, 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles.
URL http://www.sciencedirect.com/science/article/pii/S1474667016306280

Toldo, R., Fusiello, A., 2008. Robust multiple structures estimation with j-linkage. In: European Conference on Computer Vision. pp. 537–547.

Wang, H. H., Rock, S. M., Lees, M. J., Oct 1995. Experiments in automatic retrieval of underwater objects with an auv. In: OCEANS '95. MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings. Vol. 1. pp. 366–373 vol.1.

Wikipedia contributors, 2018a. Adaptive histogram equalization — Wikipedia, the free encyclopedia. [Online; accessed 15-July-2018].
URL https://en.wikipedia.org/w/index.php?title=Adaptive_histogram_equalization&oldid=826645839

Wikipedia contributors, 2018b. Approximations of $\pi---Wikipedia,the free encyclopedia.$, [Online; accessed 15-July-2018].

Wikipedia contributors, 2018c. Automatic number-plate recognition — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Automatic_number-plate_recognition&oldid=849372475, [Online; accessed 15-July-2018].

Wikipedia contributors, 2018d. Binary image — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Binary_image&oldid=850072501, [Online; accessed 15-July-2018].

Wikipedia contributors, 2018e. Grayscale — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Grayscale&oldid=845767076, [Online; accessed 15-July-2018].

Wikipedia contributors, 2018f. Hsl and hsv — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=849963118, [Online; accessed 15-July-2018].

Wikipedia contributors, 2018g. Otsu's method — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Otsu%27s_method&oldid=844681806, [Online; accessed 15-July-2018].

Wikipedia contributors, 2018h. Rgb color model — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=RGB_color_model&oldid=849447322, [Online; accessed 15-July-2018].

Youakim, D., Ridao, P., Palomeras, N., Spadafora, F., Romags, D., Muzzupappa, M., 04 2017. Autonomous underwater free-floating manipulation using moveit! PP, 1–1.

Zuiderveld, K., 1994. Contrast limited adaptive histogram equalization. Academic Press Professional, Inc.

Appendix

Image Pre-processing

```
def image_preprocessing(frame):
        HSVframe = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        chans = cv2.split(HSVframe)#
        grayframe = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        (r,c,k)= frame.shape
        return chans,grayframe,r,c
```

Removal of interfere information in camera image

```
def label_remove_ll(chans,m,n,imshow = 0):
        # yellow label & white label
        im_yellow = cv2.subtract(chans[1],chans[0])
        im_white = cv2.subtract(chans[2],chans[1])
        # all label
        ma = cv2.add(im_yellow,im_white)
        (ret,th_ma) = cv2.threshold(ma,200,255,cv2.THRESH_BINARY)
        # expand label area
        erode_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
        (80,80))
        mask = cv2.dilate(th_ma,erode_kernel)
        # frame surrounding
        # (m,n) = mask.shape
        cv2.rectangle(mask, (0,10),(n,0),(255,255,255),-1)
        cv2.rectangle(mask, (0,m),(n,m-10),(255,255,255),-1)
        cv2.rectangle(mask, (0,m),(10,0),(255,255,255),-1)
        cv2.rectangle(mask, (n-10,m),(n,0),(255,255,255),-1)
        label_mask = cv2.bitwise_not(mask)
        return label_mask
```

Line feature extraction

```
dual_canny_hough(chans,mask,imshow = 1):
        Xvs = cv2.subtract(chans[2],chans[1])
        imagevs = cv2.GaussianBlur(Xvs,(15,15),0)
        imagevscan = cv2.Canny(imagevs, 3, 40)
        imagevs = cv2.bitwise_and(imagevscan,imagevscan,
        mask = mask)
        lines = cv2.HoughLines(imagevs,1,np.pi/180,60)
        if (imshow == 1):
                cv2.imshow("1 Enhanced grayscale",Xvs)
                cv2.imshow("2 Canny",imagevscan)
                cv2.imshow("3 Canny remove",imagevs)
        if lines is not None:
                return lines
```

Removal of duplicate lines

```
def lineclass(lines):
        l = [[[0,0]]]
        l[0][0][0] = lines[0][0][0]
        l[0][0][1] = lines[0][0][1]
        for i in range(len(lines)):
                cond = 0
                conds = 0
                for lei in l:
                        if (cond == 2222):
                                continue
                        for line in lei:
                                if (cond == 1111):
                                        continue
                                else:
                                        t_lines = lines[i][0][1]
                                        t_l = line[1]
                                        r_lines = lines[i][0][0]
                                        r_l = line[0]
                                        theta_d = abs(t_lines-t_l)
                                        if (theta_d <=0.25) or
                                        (2.89<=theta_d <=3.39):
                                                rho_d =
                                                abs(r_lines-r_l)
                                                if (rho_d <=40):
```

```
                                                        line[0] = 0.5*
                                                        (r_lines+r_l)
                                                        line[1] = 0.5*
                                                        (t_lines+t_l)
                                                        cond = 1111
                                                else:
                                                        cond = 2222
                                        else:
                                                conds = 3333
                        if (cond == 2222):
                        lei.append([lines[i][0][0],lines[i][0][1]])
                if (cond == 0):
                        if (conds == 3333):
                                l.append([[lines[i][0][0],
                                lines[i][0][1]]])
        return l
```

Drawing lines extracted by Hough Transformation

```
def lgcore_HoughLineCalculation(cos,sin,theta,rho):
        x0 = cos*rho
        y0 = sin*rho
        x1 = int(x0 + 100000*(-sin))
        y1 = int(y0 + 100000*(cos))
        x2 = int(x0 - 100000*(-sin))
        y2 = int(y0 - 100000*(cos))
        return x1,y1,x2,y2
```

Line classification

```
def lgcore(l,origin,r,c,imshow = 1,linelist = [0,0,0,0,0,0,0,0,0,0]):
        yaw_l = [1,0,0,0,0,0,0,0]
        yaw_r = [1,0,0,0,0,0,0,0]
        ytransl = [0.707,(0,0),(0,0),(0,0),0,0]
        for lei in range(len(l)):
                theta = l[lei][0][1]
                rho = l[lei][0][0]
                cos = np.cos(theta)
```

```
sin = np.sin(theta)
if (0<=cos<=0.707 and 0.707<sin<=1):
        if (linelist[0] == 1):
                x1,y1,x2,y2 =
                lgcore_HoughLineCalculation
                (cos,theta,rho)
                cv2.line(origin,(x1,y1),(x2,y2),
                (255,0,0),1)
        if (cos < yaw_r[0]):
                yaw_r[0] = cos
                yaw_r[1] = theta
                yaw_r[2] = rho
                yaw_r[3] = cos
                yaw_r[4] = sin
                yaw_r[5] = 2
                yaw_r[6] = 1.57 - theta
if (-0.707<=cos<=0 and 0.707<sin<=1):
        if (linelist[1] == 1):
                x1,y1,x2,y2 =
                lgcore_HoughLineCalculation
                (cos,theta,rho)
                cv2.line(origin,(x1,y1),(x2,y2),
                (255,255,255),2)
        if (0 - cos < yaw_l[0]):
                yaw_r[0] = 0 - cos
                yaw_r[1] = theta
                yaw_r[2] = rho
                yaw_r[3] = cos
                yaw_r[4] = sin
                yaw_r[5] = 1
                yaw_r[6] = theta - 1.57
if (0<=cos<=0.707 and -1<=sin<-0.707):
        if (linelist[2] == 1):
                x1,y1,x2,y2 =
                lgcore_HoughLineCalculation
                (cos,theta,rho)
                cv2.line(origin,(x1,y1),(x2,y2),
                (0,255,0),1)
        if (cos < yaw_l[0]):
                yaw_r[0] = cos
                yaw_r[1] = theta
                yaw_r[2] = rho
                yaw_r[3] = cos
                yaw_r[4] = sin
                yaw_r[5] = 1
```

```
                    yaw_r[6] = theta - 4.71
        if (0.707<cos<=1 and -0.707<=sin<=0):
                x1 = int(x0 + 100000*(-b))
                y1 = int(y0 + 100000*(a))
                x2 = int(x0 - 100000*(-b))
                y2 = int(y0 - 100000*(a))
                if (linelist[3] == 1):
                        cv2.line(origin,(x1,y1),(x2,y2),
                        (255,255,255),1)
                if (cos > ytransl[0]):
                        ytransl[0] = cos
                        x1 = rho*np.cos(theta - 4.71)
                        x1 = int(x1)
                        ytransl[1] = (x1,0)
                        x2 = x1+r*np.tan(theta - 4.71)
                        x2 = int(x2)
                        ytransl[2] = (x2,r)
                        xm = int((x1+x2)/2)
                        ym = int(r/2)
                        ytransl[3] = (xm,ym)
                        ytransl[4] = rho
                        ytransl[5] = theta
        if (0.707<cos<=1 and 0<=sin<=0.707):
                x1 = int(x0 + 100000*(-b))
                y1 = int(y0 + 100000*(a))
                x2 = int(x0 - 100000*(-b))
                y2 = int(y0 - 100000*(a))
                if (linelist[4] == 1):
                        cv2.line(origin,(x1,y1),(x2,y2),
                        (255,255,255),1)
                if (cos > ytransl[0]):
                        ytransl[0] = cos
                        x1 = rho*np.cos(theta)
                        x1 = int(x1)
                        ytransl[1] = (x1,0)
                        x2 = x1-r*np.tan(theta)
                        print("x2",x2)
                        x2 = int(x2)
                        ytransl[2] = (x2,r)
                        xm = int((x1+x2)/2)
                        ym = int(r/2)
                        ytransl[3] = (xm,ym)
                        ytransl[4] = rho
                        ytransl[5] = theta
        if (-1<=cos<-0.707 and 0<=sin<=0.707):
```

```
                        x1 = int(x0 + 100000*(-b))
                        y1 = int(y0 + 100000*(a))
                        x2 = int(x0 - 100000*(-b))
                        y2 = int(y0 - 100000*(a))
                        if (linelist[5] == 1):
                                cv2.line(origin,(x1,y1),(x2,y2),
                                (255,255,255),1)
                        if (cos > ytransl[0]):
                                ytransl[0] = cos
                                x1 = rho/np.cos(theta - 1.57)
                                x1 = int(x1)
                                ytransl[1] = (x1,0)
                                x2 = x1+r/np.cos(theta - 1.57)
                                x2 = int(x2)
                                ytransl[2] = (x2,r)
                                xm = int((x1+x2)/2)
                                ym = int(r/2)
                                ytransl[3] = (xm,ym)
                                ytransl[4] = rho
                                ytransl[5] = theta
        yaw = []
        if yaw_l[0] < yaw_r[0]:
                yaw = yaw_l
        else:
                yaw = yaw_r

        return yaw
```

Calculating and drawing the reference line

```
def reference_draw(origin,m,n,yaw):
        y_m = int(m/2)
        x_m = int(n/2)
        circle_rho = int(n/6)
        y_increment = x_m * np.tan(yaw[6])
        rho_x_increment = circle_rho * np.cos(yaw[6])
        rho_y_increment = circle_rho * np.sin(yaw[6])
        cv2.line(origin,(0,y_m),(n,y_m),(255,255,255),1)
        font = cv2.FONT_HERSHEY_SIMPLEX
        text_rad = str(round(yaw[6],3))+"rad"
        deg = yaw[6] * 180 / 3.14
        text_deg = str(round(deg,1))+"deg"
```

```
if yaw[5] == 2:
        x_ref_1 = int(x_m - rho_x_increment)
        y_ref_1 = int(y_m + rho_y_increment)
        y_ref_2 = int(y_m - y_increment)
        cv2.line(origin,(x_ref_1,y_ref_1),(n,y_ref_2),
        (0,255,255),1)
        cv2.putText(origin, text_rad ,
        (x_m - circle_rho - 150, y_m + 25), font, 1,
        (0, 255, 0), 2,False)
        cv2.putText(origin, text_deg ,
        (x_m + circle_rho, y_m), font, 1, (0, 255, 0),
        2,False)
else:
        y_ref_1 = int(y_m - y_increment)
        y_ref_2 = int(y_m + rho_y_increment)
        x_ref_2 = int(x_m + rho_x_increment)
        cv2.line(origin,(0,y_ref_1),(x_ref_2,y_ref_2),
        (0,255,255),1)
cv2.circle(origin,(x_m,y_m),circle_rho,(255,255,255),1)
cv2.line(origin,(x_m,0),(x_m, m),(255,255,255),1)
w = int(n/2)-5
# ad = int(m/2)
direction = yaw[5]
if direction is 1:#left
        arrow = np.array([[[125,y_m-22], [125,y_m],[27,y_m],
        [27,y_m-81],[5,y_m-81],[38,y_m-124],
        [71,y_m-81],[49,y_m-81],[49,y_m-22]]],
        dtype = np.int32)
elif direction is 2:#right
        arrow = np.array([[[n-125,y_m-22], [n-125,y_m],
        [n-27,y_m],[n-27,y_m-81],[n-5,y_m-81],
        [n-38,y_m-124],[n-71,y_m-81],[n-49,y_m-81],
        [n-49,y_m-22]]], dtype = np.int32)
elif direction is 3:#up
        arrow = np.array([[[w,10], [w-33,35],
        [w-11,35], [w-11,75],[w+11,75],
        [w+11,35], [w+33,35]]], dtype = np.int32)
else:#down
        arrow = np.array([[[w,m-10], [w-33,m-35],
        [w-11,m-35], [w-11,m-75],[w+11,m-75],
        [w+11,m-35],[w+33,m-35]]], dtype = np.int32)
cv2.fillPoly(origin, arrow, (0,255,0))


# if (imshow == 1):
```

```
        cv2.imshow("3 line ",origin)
```

Drawing auxiliary arrow

```
def arrow_define(m,n,direction):
        if direction is 1:#left
                arrow = np.array([[[10,ad], [35,ad+33],[35,ad+11],
                [75,ad+11],[75,ad-11],[35,ad-11],
                [35,ad-33]]], dtype = np.int32)
        elif direction is 2:#right
                arrow = np.array([[[n-10,ad], [n-35,ad+33],
                [n-35,ad+11],[n-75,ad+11],[n-75,ad-11],
                [n-35,ad-11],[n-35,ad-33]]], dtype = np.int32)
        elif direction is 3:#up
                arrow = np.array([[[w,10], [w-33,35],
                [w-11,35], [w-11,75],[w+11,75],
                [w+11,35],[w+33,35]]], dtype = np.int32)
        else:
                arrow = np.array([[[w,m-10], [w-33,m-35],
                [w-11,m-35], [w-11,m-75],[w+11,m-75],
                [w+11,m-35],[w+33,m-35]]], dtype = np.int32)
        return arrow
```

Drawing auxiliary arrow

```
def LabelDetect(HSVframe, grayframe, chans, mask,
    dilate_kernel, shadow_kernel, imshow = 1):
        closed = cv2.morphologyEx(grayframe, cv2.MORPH_CLOSE
        , shadow_kernel)
        removeshadow = cv2.subtract(closed, chans[2])
        removeshadowcut = cv2.bitwise_and(removeshadow,
        removeshadow, mask = mask)
        equal_imgcut = cv2.equalizeHist(removeshadowcut)
        ll = 47
        blur = cv2.GaussianBlur(equal_imgcut,(ll, ll),0)
        T1 = mahotas.thresholding.otsu(blur)
        thresh = blur.copy()
        thresh[thresh > T1] = 255
        thresh[thresh < 255] = 0
        label11 = cv2.bitwise_and(equal_imgcut, equal_imgcut,
        mask = thresh)
        (_, lcnts, _) = cv2.findContours(thresh.copy(),
        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)#SIMPLE)

        if imshow == 1:
                cv2.imshow("1 valveframe", valveframe)
                cv2.imshow("2 Dilate Inverse", dilatednot)
                cv2.imshow("3 Close operation", closed)
                cv2.imshow("4 Open operation", opened)
                cv2.imshow("4 OpenCV subtract", removeshadow)
                cv2.imshow("5 removecut", removeshadowcut)
                cv2.imshow("6 clahe cut", equal_imgcut)
                cv2.imshow("7 GaussianBlur", blur)
                cv2.imshow("8 Ostu's thresh", thresh)
                cv2.imshow("9 label11", label11)
                cv2.imshow("10 label segmentation", label11)

        return lcnts
```