CT07 - OBSTACLE DETECTION AND AVOIDANCE FOR AN UNMANNED SURFACE VEHICLE (USV)

Submitted by

Pranay Shirodkar

A0124257J

Department of

Mechanical Engineering

In partial fulfilment of the

requirements for the Degree of

Bachelor of Engineering

National University of Singapore

Session 2016/2017

Summary

This Final Year Project achieves Obstacle Detection for an Unmanned Surface Vehicle (USV), enabling it to operate independently on the water surface. The proposed obstacle detection algorithm combines stereo vision and point cloud processing techniques to identify obstacles in the USV's environment and determine the position of these obstacles in relation to the USV. This information is crucial for the USV to execute autonomous navigation safely.

The obstacles are detected using two cameras mounted facing forward on the USV. The image streams from these two cameras serve as the inputs to the obstacle detection algorithm. Stereo vision is the first phase of the obstacle detection algorithm. To set up the two cameras for stereo vision, camera calibration is conducted and the necessary camera parameters acquired. Although this process is conducted only once, it is a critical step in ensuring accurate stereo vision results. Using the camera calibration data, the two image streams are rectified and processed by a stereo block matching algorithm. This algorithm generates a disparity map that contains depth information about the images. With this depth information, the disparity map is transformed into a 3D point cloud for point cloud processing.

Point cloud processing is the second phase of the obstacle detection algorithm. Using a plane segmentation algorithm, the plane of points that represent the water surface is extracted from the point cloud, leaving only obstacles behind. As a post processing step, a statistical outlier filter is applied to remove any noise detected in the point cloud. The information that remains is the critical point cloud data that

represents obstacles in the USV's environment. This information is used to generate a 2D obstacle map, which is the end result of the entire obstacle detection algorithm. This map can subsequently be used by the USV in path planning and navigation.

The algorithms were coded in C++ programming language on the Robot Operating System (ROS) platform to allow seamless integration with all other software and sensors present on the USV. The frequently used libraries include OpenCV and Point Cloud Library (PCL).

Experiments were conducted to test the effectiveness of the obstacle detection algorithm and the obstacle map that it generates. The algorithm functioned well in both an indoor and an outdoor environment. The results validate that the obstacle detection algorithm is effective in detecting obstacles in the USV's surroundings.

Recommendations to make further improvements to the algorithms are suggested at the end of this report. The stereo vision phase can be enhanced by improving the disparity map generated by the block matching algorithm. Pre-processing the camera images or post-processing the disparity map could achieve this. Using a higher resolution camera and conducting more precise camera calibration would also improve the accuracy of the disparity maps. Enhancing the plane segmentation algorithm is another area for improvement. It is possible that the implemented segmentation algorithm is unable to differentiate between the water surface and low lying obstacles. As such, other segmentation algorithms should be explored. Ensuring the effectiveness of the algorithm in all possible situations would ensure highly accurate obstacle maps for safe USV navigation.

Table of Contents

List of Figures

List of Symbols

c_x, c_y	Coordinates of the principal point in the left camera
c'_x, c'_y	Coordinates of the principal point in the right camera
D	Distortion Matrix
$disp(x, y)$	Disparity value at (x, y) in left camera image
f	Camera focal length
K	Camera Matrix or Intrinsic Matrix
P	Projection Matrix
Q	Disparity-to-depth mapping Matrix
R	Rectification Matrix
$[Rot\|t]$	Extrinsic Matrix, consisting of rotation and translation matrix
s	Skew of camera focal axis
T	Distance from the optical centre of one camera to the other
W	Scale factor
x, y	Coordinates in image coordinate frame
X, Y, Z	Coordinates in point cloud coordinate frame, before scaling

1 Introduction

Recent technological advancements have enabled the development of Unmanned Surface Vehicles (USVs), autonomous crafts that operate on the water surface without on-board crew. With autonomous or offshore control capabilities, USVs are becoming increasingly relevant and important to oceanographers who wish to study the ocean, or to military personnel conducting maritime surveillance and reconnaissance activities [1]. For a USV to operate autonomously, it is critical that it can sense obstacles in its surroundings and safely navigate around them. This would avoid damage to the structure of the USV and ensure that it has a long operational life. This serves as the motivation for this final year project (FYP).

Figure 1: Team Sharky, NUS Unmanned Surface Vehicle

1.1 Objective

The objective of this FYP is to develop the necessary software and algorithms that would equip the USV with the capability to autonomously navigate on a body of water to its destination without colliding into any obstacles.

1.2 Problem

Obstacle detection and avoidance is essential for a USV to function independently at sea. The algorithm should be robust, enabling the USV to cope with obstacles in harsh sea environments and also integrate well with on board complementary equipment and software. Implementing such a system would require the following stages:

- Detecting obstacles in the USV's surroundings
- Mapping the position of these obstacles relative to the USV
- Determining a route to the destination that safely avoids all obstacles

1.3 Scope

This cumulative FYP continues on the development of the project where the previous student discontinued. The task is to improve on the existing system as effectively as possible, rather than develop the system from the beginning.

One major area for improvement is in detecting obstacles in the USV's surroundings. The current method is to rely on LIDAR (Light Detection and Ranging) technology. Although the LIDAR worked well during the indoor experiment, it detected a lot of noise and false positives during the outdoor

experiment. The performance of the obstacle detection algorithm was insufficient and needed to be refined.

As a result, the objective of this project is narrowed to improve in particular, the obstacle detection capabilities of the USV. Extensive research is conducted into obstacle detection sensors and their corresponding detection and mapping techniques. Possible approaches to the problem are discussed before determining the selected obstacle detection methodology for the project. The methodology is the core of this project, and it is covered in the following order: camera calibration, stereo vision, and point cloud processing. Where appropriate, theory is introduced and algorithms are explained.

In the results section, the experimental tests of the software are analysed and shown to validate the effectiveness of the algorithms indoors and outdoors. Lastly, recommendations for further improvement are discussed to conclude this project.

2 Literature Survey

A literature review was prepared to determine the best way to improve the obstacle detection capability of the USV.

One of the frequently used sensors for obstacle distance determination is the LIDAR discussed earlier. The LIDAR utilised by the previous student is the SICK LMS-151. The instrument fires rapid pulses of laser and calculates the distance and bearing of an obstacle based on the time taken for the each pulse to be reflected. Unfortunately, during experimentation performed previously for this project, it was

found that the results of the LIDAR were ineffective when dealing with outdoor environments due to the presence of false positives being detected. The noise makes this obstacle detection method ineffective.

Alternatively, computer vision is a commonly used tool to provide robots with image processing capabilities including acquiring, processing, analysing and understanding digital images, and extracting higher-dimensional data about the real world in order to produce useful information for decision making. Using cameras and computer vision techniques, it is possible to perceive the position of obstacles around the USV and navigate around them safely. In particular, a two camera stereo vision system can enable the position of obstacles to be perceived.

Furthermore, it was found that LIDAR can be affected by the reflectivity of the water surface and is limited to an obstacle detection range of approximately 15 metres. In contrast, the stereo vision detection range can go up to 20 metres or greater with well calibrated high resolution cameras [2]. Stereo vision has also been reliably tested and implemented in a number of projects for similar applications by the research community [3]. As a result, these factors justify the usage of cameras as the optimal sensor for accurate obstacle detection in the USV's environment.

3 Methodology

In this chapter, the obstacle detection algorithm employed will be broken down and presented in the following order: 1) Camera Calibration, 2) Stereo Vision

Algorithm, and 3) Point Cloud Processing [4]. In each section, the approach, techniques and algorithms will be discussed and explained.

3.1 Overview of the Methodology

1. The raw image captured by a camera may have radial, tangential or fish-eye lens distortion characteristics that are intrinsic to the camera. Camera Calibration must be performed to determine the important camera setup parameters needed to define these errors. Using the information from camera calibration, images are undistorted to remove errors, ensuring that the images agree with the pinhole camera projection model. The image must also be rectified to facilitate comparison between images in the next step.
2. A stereo vision algorithm is implemented to match features that are present in both images. The algorithm minimises an information measure that is used to compare different regions between image pairs. This gives the best estimate of the position of a common feature in both images. The difference in position of the common feature in the two images is encoded into a disparity map. This disparity map contains depth information and is projected into a 3D point cloud. Using camera calibration parameters, the point cloud is computed to provide 3D information about the real world.
3. In the point cloud processing stage, a plane segmentation algorithm is implemented. This algorithm can eliminate the portion of the point cloud that represents the water surface, as this is not an obstacle. Finally, the remaining information is used to determine the position of the obstacles in the 3D point cloud and this information is projected onto a 2D image map for the USV's navigational purposes.

3.2 Obstacle Detection Algorithm

3.2.1 Camera Calibration

Camera calibration estimates the geometric parameters of a lens and image sensor belonging to a video camera. These parameters are used to correct for lens distortions and to rectify images [5]. To determine the camera parameters, one needs a 3D world object with known dimensions and its corresponding position on the 2D image seen by the camera. The object with known dimensions designed for this project is a checkerboard, a frequently used camera calibration tool. With this checkerboard, existing camera calibration packages are employed to generate the required camera parameters.

3.2.1.1 Camera setup and calibration

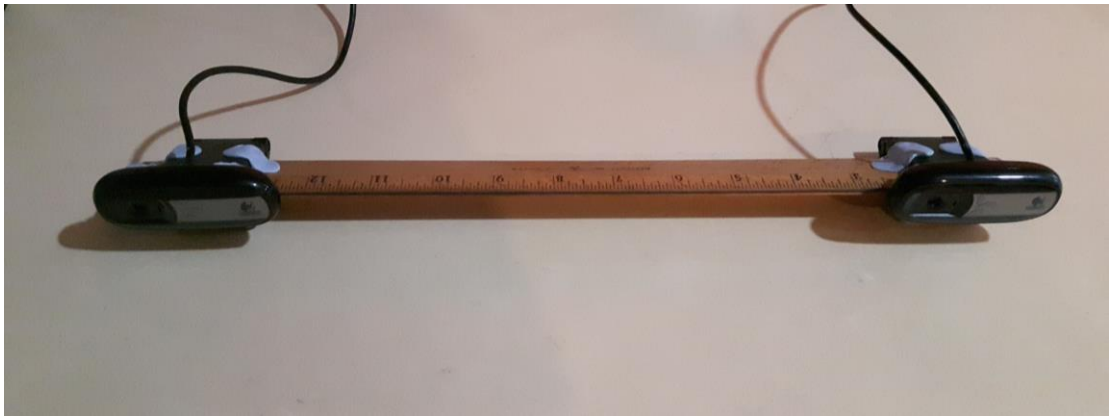

Figure 2: Camera setup of two C170 Logitech Webcams

As shown in Figure 2, the cameras have been firmly mounted onto a wooden ruler. It is important to mount the cameras on a stiff structure to ensure that the position of the cameras relative to one another does not change throughout this project. This is to minimise the need for camera re-calibration.

The cameras were calibrated using the ROS Camera Calibration package [6]. The checkerboard used had an individual square length of 70mm, with a size of 8x7, counting interior corners, (the standard chessboards has 8x8 squares, but 7x7 interiors corners) which is the feature detected by the camera calibration package. The images from the webcams were converted to grayscale before calibration for compatibility with the package.

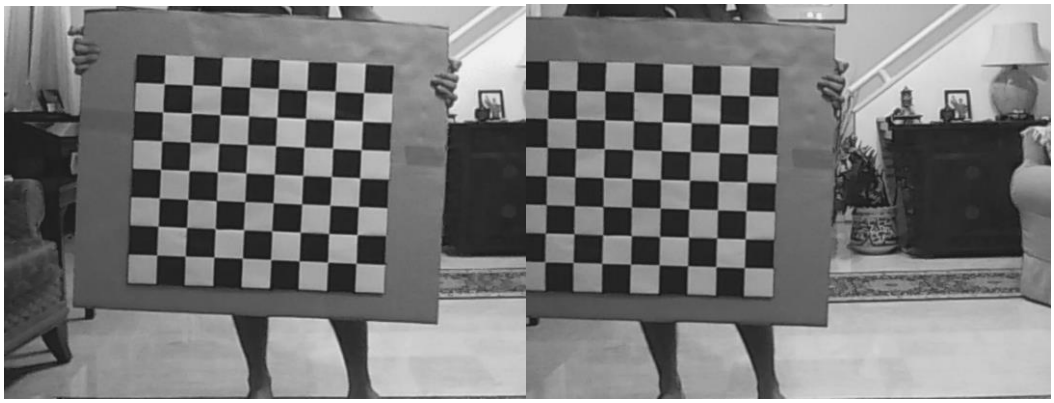

Figure 3: Left image and Right image for one calibration sample

Using 44 pairs of calibration sample images, the package generated 4 matrices that quantify the camera parameters: camera_matrix (K), distortion_coefficients (D), rectification_matrix (R) and projection_matrix (P). The parameter values used can be found in Appendix 1.

3.2.1.2 Camera parameters

The D matrix corrects lens distortion errors that are a limitation of the camera. The R matrix rectifies the image so that comparison between image pairs can be performed more effectively.

$$K = \begin{pmatrix} f & s & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

$$P = K \times [Rot|t] \quad (2)$$

The K matrix represents the geometric properties of the camera, such as focal length, position of the focal axis and skew. The 3x4 P matrix represents a transformation matrix that maps coordinates from 3D to 2D [7]. All these relationships are represented in the form of Equations (1) and (2). As matrices K and P have been obtained from the camera calibration package, they are used to solve for the extrinsic matrix $[Rot|t]$. This extrinsic matrix is a 3x3 rotation matrix in the left-block, and a 3x1 translation column-vector in the right-block. The extrinsic matrix describes the camera's location in the world, and the direction it is facing. The matrices P and K are important parameters that enable the subsequent stereo vision algorithm to function.

3.2.2 Stereo Vision Algorithm

In stereo vision, two cameras are horizontally displaced and the two image streams provide differing views of the same scene, akin to the role that eyes play in human binocular vision. Using a stereo vision algorithm, such as the block matching algorithm, these two images are analysed and used to generate a disparity map. A disparity map contains information about the depth of image features that are present in both camera images. The values in this disparity map are inversely proportional to the scene depth at that pixel location.

3.2.2.1 Block Matching Algorithm

The block matching (BM) algorithm is a frequently used technique to achieve stereo vision and is used extensively by the computer vision community. It exploits the fact that when setting up two horizontally displaced cameras, objects in the cameras' field of view will appear at slightly different locations within the two images due to the cameras' different perspectives of the scene and how far the object is from the cameras.

Figure 4: Left rectified and Right rectified images used for Stereo Block Matching Algorithm

Features closer to the cameras have a larger difference in location between the two images compared to features further away from the camera. As Figure 5 shows, the bag, which is closer to the cameras, has a large difference in position between the left and right image. The cupboard in the background is further away, and thus shows a small difference in position between the left and right image. This difference in the position of an object or feature is called the disparity. It is defined as the difference, measured in pixels, of the position of the feature in the left image

and its corresponding position in the right image. Computing the disparity for every feature in the image pair results in a disparity map.

However, finding the disparity for a feature first requires the ability to recognise the same feature in the left and right image. This is achieved through block matching [8]. Beginning with a small region of pixels that contains the feature of interest in the left image, a search is conducted for the closest matching region of pixels in the right image. The search in the right image begins at the same position as the feature in the left image and proceeds horizontally to either side. The similarity of the regions is determined by calculating their sum of absolute differences (SAD). This involves adding up all the differences in pixel values between the region of interest from the left image and the region being considered for similarity in the right image. The SAD value is computed for every region in the right image that is being considered for similarity. The region with the smallest SAD value will be the region that matches the initial feature of interest in the left image that was being investigated. Now that the same feature has been recognised in the left and right image, their pixel distance or disparity can be found.

The BM algorithm is an appropriate algorithm choice for implementation in this project because it is a very fast stereo algorithm. While some algorithms like the graph cut algorithm can take minutes to process one pair of stereo images, the BM algorithm can process stereo image pairs rapidly enough such that it is effective in real time situations. Although it compromises on its accuracy in matching features between the left and right images, it is able to produce disparity maps that are suitable and perform sufficiently well for this implementation.

3.2.2.2 Block Matching Algorithm parameter tuning

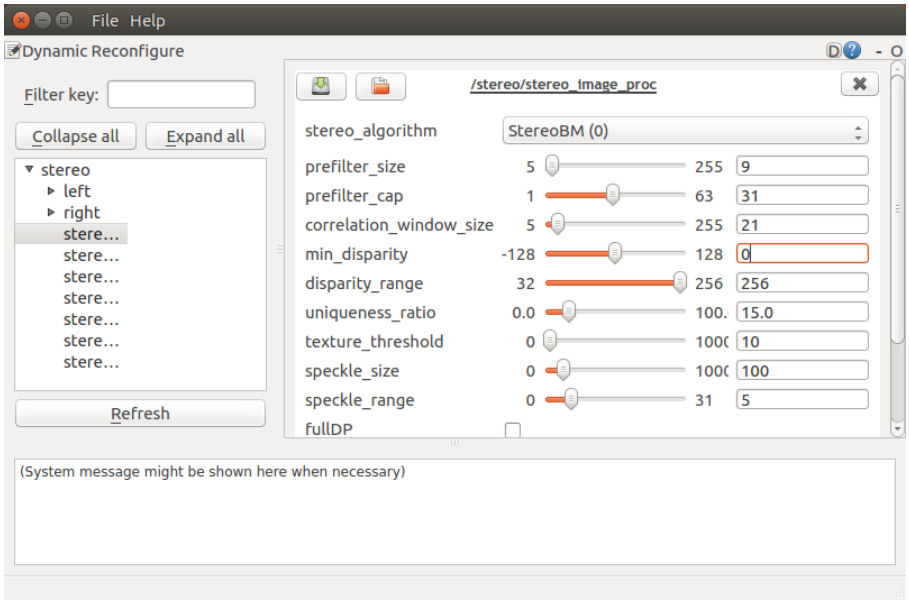

Figure 5: Block Matching Algorithm parameters

Tuning the BM Algorithm parameters is an important step in ensuring that the algorithm matches features successfully [9]. The tuning is also specific to the intended range of detection. For the purpose of indoor feature mapping with the stereo camera setup, the estimated detection range necessary is 0 - 6 metres. This range can subsequently be adjusted for outdoor stereo imaging.

The range is controlled by manipulating the min_disparity and disparity_range parameters. Since the cameras are positioned 28 centimetres apart, the disparity values expected for features that are near the camera is quite large. Thus, a high disparity_range is required to detect these features. By setting min_disparity and disparity_range to 0 and 256 respectively, the detection range is set to 0 - 6 metres for indoor usage. By constraining the detection range to 0 - 6 metres, better depth resolution is attained within this range. To further improve the results, the correlation_window_size is increased to 21. The correlation_window_size refers to

the region of pixels containing the feature of interest in the left image, as discussed in 3.2.2.1. Increasing its size would increase the accuracy of finding the corresponding feature in the right image correctly.

3.2.2.3 Block Matching Algorithm implementation results

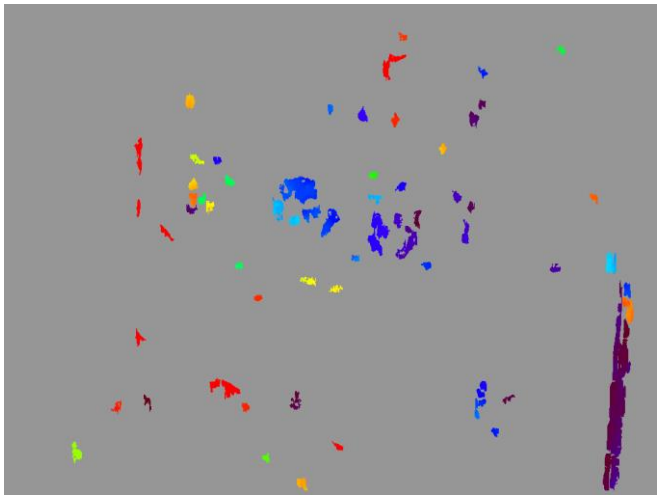

Figure 6: Disparity image before parameter tuning

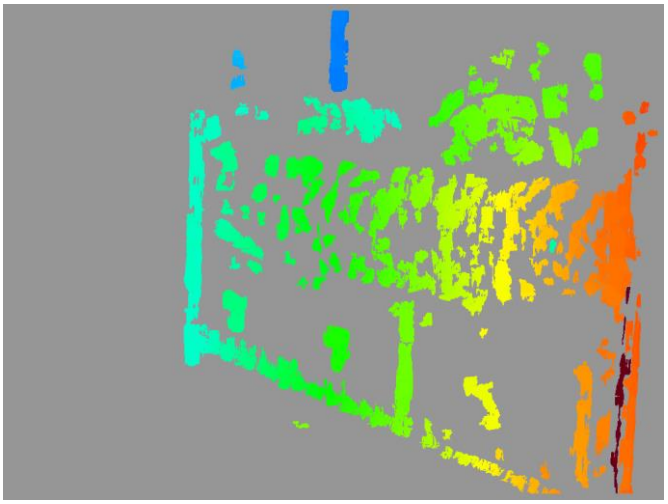

Figure 7: Disparity image after parameter tuning

Before tuning the BM Algorithm parameters, the disparity map result is very poor and filled with noise, as demonstrated by Figure 6. Tuning is a very important step

that is carried out prior to operating the USV, depending on the range of detection desired. After tuning, the disparity map in Figure 7 is obtained. This disparity map is interpreted by comparing it with the left camera image in Figure 4. On a rainbow spectrum of colours (ROYGBIV), the largest disparities in Figure 7 are represented in red and the lowest disparities are represented in violet. Comparing the left camera image in Figure 4 and the disparity map in Figure 7 shows that the end of the bed closer to the cameras has higher disparity values (orange in colour) while the other end of the bed is further from the cameras and thus has lower disparity values (light blue-green in colour). The shape of the bed, pillows and bag on the bed are also well detected in the disparity image. This result validates that the algorithm is functioning well and can effectively extract the important depth information from the scene.

The grey sections of the disparity image are the regions for which no feature mapping could be found. This is generally because, the algorithm performs feature mapping poorly for untextured surface. Consider a particular region of pixels on the white wall in the left image. It would be difficult for the BM Algorithm to determine which region of pixels in the right image corresponds to this region of pixels in the left image, because there are many regions in the right image consisting entirely of the white wall and they would all be suitable candidates. It is clear that the algorithm does not perform well for untextured regions such as the white wall, the cupboard doors, and the floor. Thus, they are unmapped and grey in the disparity image. In contrast, the bed has good texture and this enables the algorithm to perform feature mapping well.

Unfortunately, utilising a more robust stereo algorithm that handles untextured features would be too computationally expensive and time consuming for this project. The implemented algorithm in this project must perform real time obstacle detection for the USV to perform safe navigation on the water surface. As such, this highlights the importance of performing highly accurate camera calibration, and tuning the BM parameters carefully to maximise the effectiveness of the BM algorithm in generating accurate disparity information from the camera images.

3.2.3 Point Cloud Processing

A point cloud is a collection of data points which have been defined in a particular coordinate system. In a three-dimensional coordinate system, these points are defined by 3 coordinates, and often are intended to represent the external surface of an object. This section of the report discusses point cloud processing and will rely on the point cloud library (PCL) and point cloud processing algorithms for support. The first component in this section is to translate the disparity map into a 3D point cloud. Subsequently, a plane segmentation algorithm is proposed to segment this 3D point cloud, removing the points in the cloud that constitute the ground or the water surface, which is not an obstacle. Thereafter, a statistical filter is applied on the point cloud to reduce the noise and extraneous information that does not represent an obstacle. Finally, only pertinent information about the obstacles in the USV's surroundings remain in the point cloud. This information is projected onto a 2D obstacle map that the USV can use in path planning operations.

3.2.3.1 Point Cloud Generation from the disparity map

$$Q = \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T & (c_x - c'_x)/T \end{pmatrix} \quad (3)$$

To transform the disparity map into a 3D point cloud, some matrix manipulation is necessary. T is called the baseline and it refers to the distance from the optical centre of one camera to the other camera. All the information required to generate the matrix Q, according to Equation (3), is available from previously determined camera calibration parameters [4].

$$[X \ Y \ Z \ W]^T = Q \times [x \ y \ disp(x,y) \ 1]^T \quad (4)$$

$$Point\ Cloud\ coordinates = [X/W, \quad Y/W, \quad Z/W] \quad (5)$$

Combining the Q matrix with the information in the disparity map and executing Equation (4) and (5) projects the disparity map into the 3D domain [4].

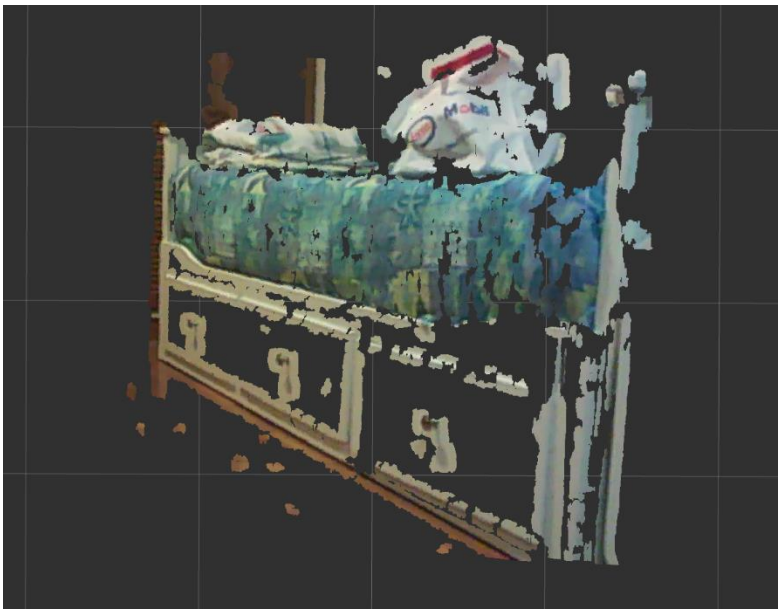

Figure 8: Unprocessed point cloud

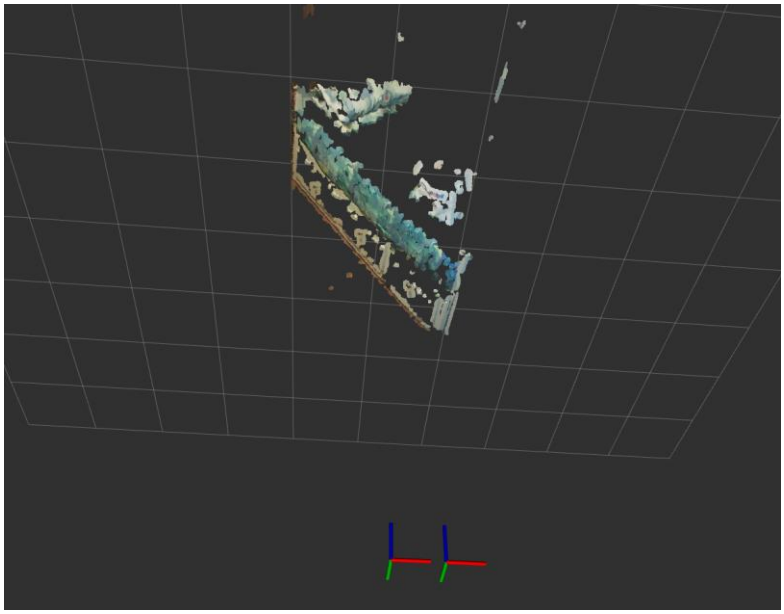

Figure 9: Point cloud viewed from a skew angle to perceive the point cloud depth

Converting the disparity map into a point cloud yielded the results visible in Figure 8 and 9. These images were obtained by using rviz, a 3D visualisation tool, which is part of ROS. Figure 8 shows the unprocessed point cloud from the perspective of the cameras. At the bottom of figure 9, the frames attached to the left and right cameras are visible. The cameras are facing the point cloud seen in Figure 9, along the blue axis. Figure 9 shows that the pattern observed from the disparity map is still present and visible in the point cloud. One end of the bed is closer to the cameras than the other end. This agrees with the results of the disparity map and validates that the obstacle detection algorithm is effective thus far.

3.2.3.2 Plane Segmentation Algorithm

In live USV operation, the cameras mounted on the USV will be processing images that contain the water surface; consisting of waves, wakes, disturbances, discolouration and foam. All these water surface features will be observed by the camera images and represented in the point cloud. However, the water surface is

not an obstacle, and it should be removed from the point cloud before the point cloud is used to generate an obstacle map. Since the water surface is represented approximately by a horizontal plane in the point cloud, an effective approach would be to identify this plane in the point cloud, and remove all points that are associated with it. This plane segmentation approach is achieved by using the random sample consensus (RANSAC) algorithm.

The RANSAC algorithm is an iterative process that estimates the parameters of a mathematical model [10], [11]. It is utilised for fitting a model to a set of data when outliers of the model have no influence on the estimated model parameter values. Equation (6) is the mathematical model used in this implementation, the equation of a plane in 3D space.

$$ax + by + cz = d \tag{6}$$

The objective of the RANSAC algorithm is to determine the values of a, b, c and d, thus determining the equation of the plane that contains the most point cloud points [10], [11]. The algorithm involves the following five steps:

1. Randomly select a subset of the original data.
2. Fit a planar model to this random subset of data.
3. Test all remaining data against this fitted planar model. Points that fit the estimated model well, are considered inliers and part of the consensus set.
4. The estimated model parameters are accurate if a sufficient proportion of points have been classified as part of the consensus set.
5. The above 4 steps count as one iteration, and a fixed number of iterations are executed. Each iteration either generates a model which is rejected because

there are too few points classified in the consensus set, or generates a model which is accepted due to its' sufficient consensus set size. In the latter case, the accepted model is kept if its consensus set is larger than the consensus set of the previously saved model.

RANSAC finds the plane with the most number of point cloud points, and during live USV operation, the water surface perceived by the camera will be translated into the largest collection of point cloud data that lies on a common plane. As such, the RANSAC algorithm is an appropriate algorithm choice to fit an approximately horizontal plane to the 3D point cloud data that represents the water surface. The algorithm is also appropriate because the points that are not part of the water surface should have no influence on the estimated model of the plane, and this is reflected in the steps of RANSAC.

The RANSAC algorithm outputs the equation of the plane representing the water surface. Points on or near enough to this plane are considered inliers and are removed from the original point cloud. The outliers, or points that are not part of the plane are retained. In essence, this step filters away the point cloud data that represented the water surface, leaving behind only information about the obstacles on the water surface. The filtered point cloud is now closer to the ground truth information about the position of obstacles around the USV.

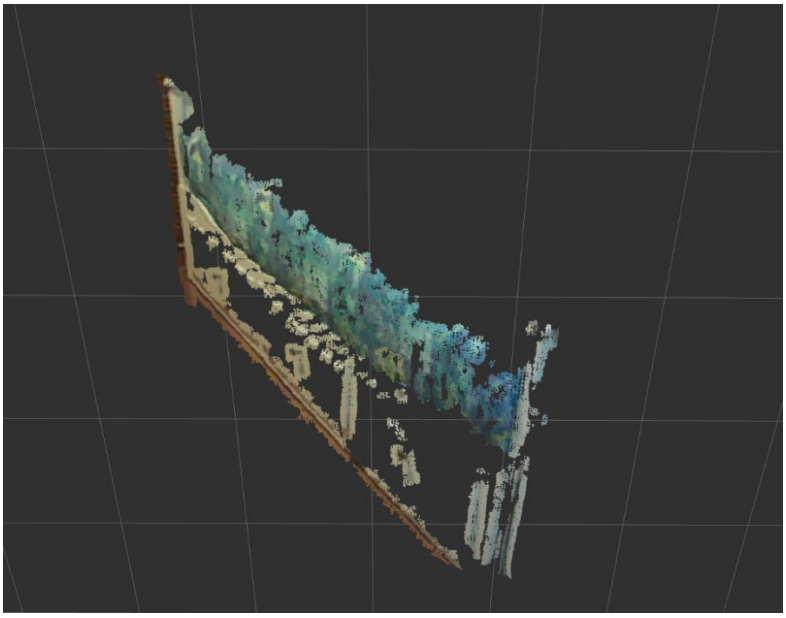

Figure 10: Plane Segmentation inliers

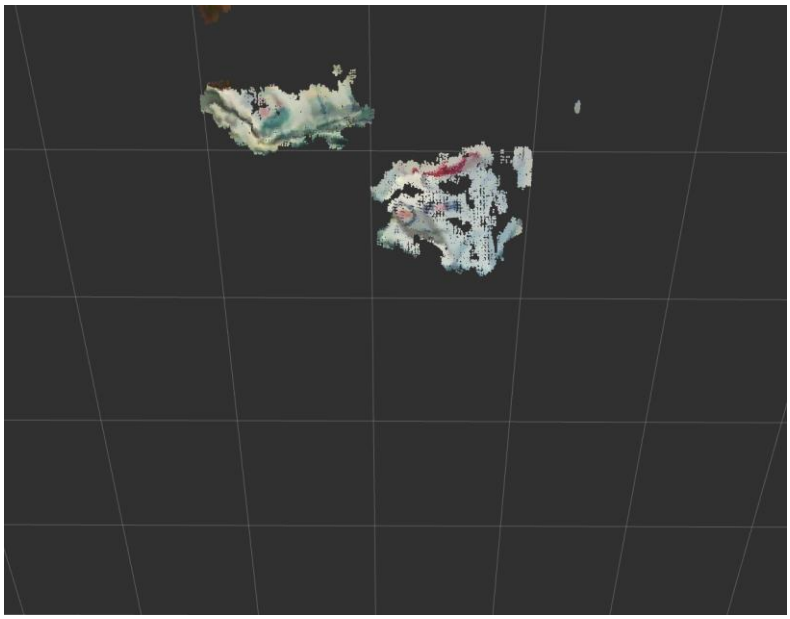

Figure 11: Plane Segmentation outliers

Applying the RANSAC algorithm for planar segmentation to the point cloud of the bedroom produces the results shown in Figure 10 and 11. The point cloud of the bedroom did not include the ground plane, due to the poor texture of the floor in the camera images. Instead, the algorithm determined the external surface of the bed

and bedframe as the plane with the maximum point cloud data. This plane was well extracted in Figure 10, leaving behind a point cloud of the objects seen in Figure 11; the external surfaces of the pillows and bag that are facing the camera. Although the bed and bedframe do not constitute an actual horizontal ground plane, the principle and proof of concept has been demonstrated. The RANSAC algorithm is effective is achieving plane segmentation for an indoor environment, as seen from the result in Figure 11.

In section 4.3 of this report, extraction of an actual ground plane is performed for an outdoor scene and the results are discussed.

3.2.3.3 Statistical Outlier Filter

The result of the obstacle detection algorithm thus far is robust but still susceptible to noise and false positives that corrupt the point cloud data. For instance, it is possible that the plane segmentation step is incomplete and the plane is not completely removed from the point cloud. Patches of the plane would riddle the remaining point cloud with noise that makes it ineffective when translated into an obstacle map. To counter this problem, a statistical outlier filter is applied on the point cloud. This will reduce errors left behind by the plane segmentation algorithm and increase the effectiveness of the obstacle detection process in generating accurate obstacle maps.

3.2.3.4 Projection to 2D Obstacle map

The last step in the obstacle detection algorithm is to translate the 3D point cloud information that remains onto an obstacle map that the USV can use to traverse its' environment safely. This is accomplished by a planar projection of all remaining

points in the point cloud onto a horizontal plane. Since plane segmentation and filtering have been conducted, the map that is generated only displays information about the obstacles and does not confuse this with the ground or with miscellaneous noise. This horizontal plane serves as the obstacle map which holds information about the dimensions and boundaries of the obstacles surrounding the USV, as perceived from the mounted cameras.

4 Results and Discussion

4.1 Overview of Algorithm with Rqt_graph

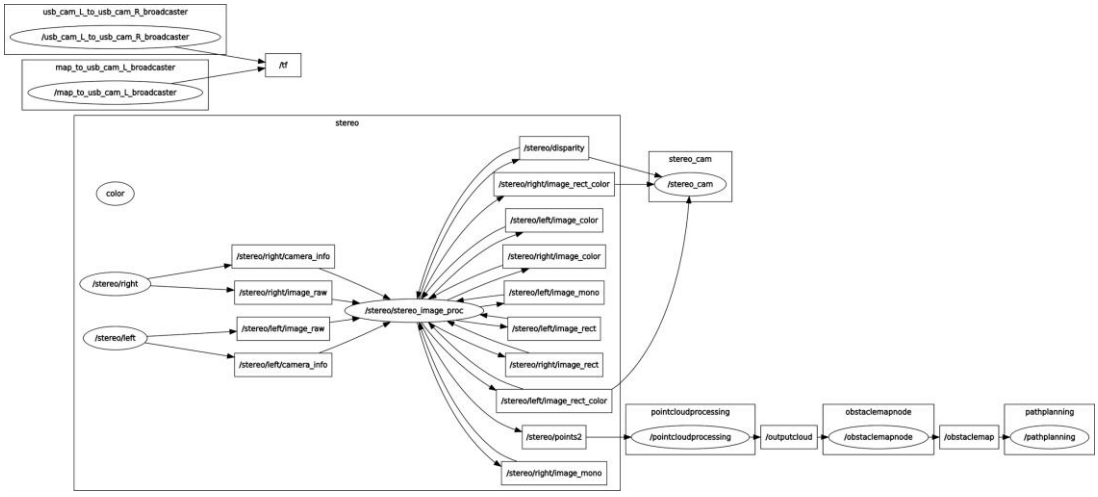

Figure 12: Rqt_graph of the obstacle detection algorithm

The overview of the obstacle detection algorithm is visualised in terms of ROS nodes and topics by referring to Figure 12. Starting from the left side of the Figure, the raw image inputs being obtained by the left and right cameras are "/stereo/left/image_raw" and "/stereo/right/image_raw". These images, together with the camera calibration data information, are used by the stereo image

processing node “/stereo_image_proc”. The stereo image processing node executes the block matching algorithm, and generates the disparity map topic “/stereo/disparity”. Subsequently, it uses the equations in 3.2.3.1 to generate the 3D unprocessed point cloud from the disparity map. The point cloud processing is carried out in the “/pointcloudprocessing” node. This node generates a 3D point cloud topic called “/outputcloud”, which is the processed point cloud at the end of section 3.2.3.3. Finally, this topic is used to generate the 2D obstacle map.

4.2 Results and discussion for indoor obstacle detection

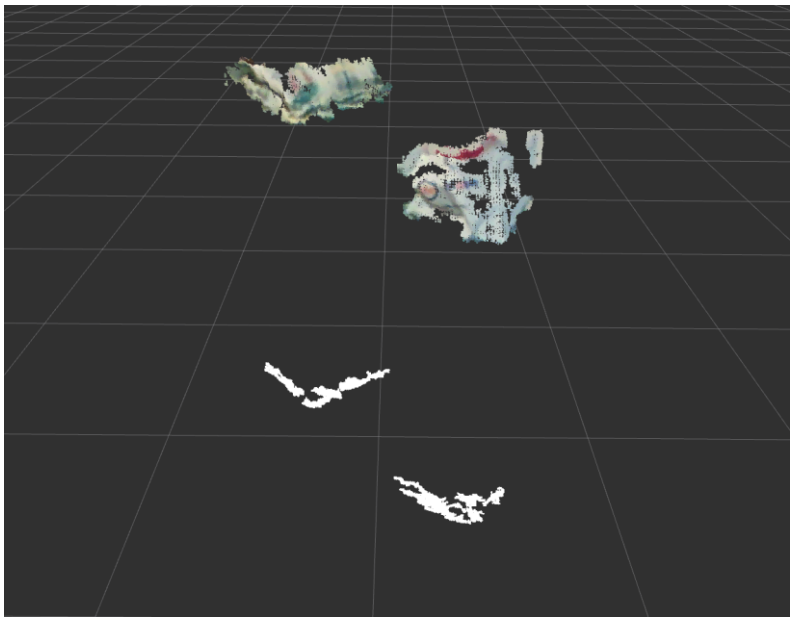

Figure 13: Projection of processed point cloud onto horizontal plane

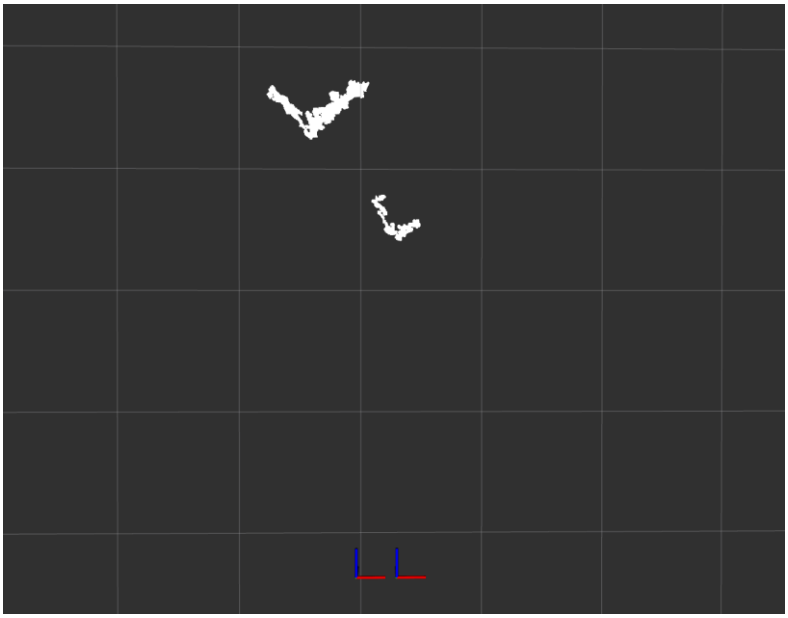

Figure 14: Obstacle map for indoor obstacle detection

Figure 13 illustrates how the processed point cloud output by section 3.2.3.3 is projected onto a horizontal plane. Figure 14 shows the obstacle map which is the final result. The white region represents the obstacles, in this case the boundaries of the pillows and the bag. At the bottom of Figure 14, the coordinate frames of the left and right cameras can be seen. The obstacle boundary of the bag is closer to the cameras than the obstacle boundary of the pillows, as expected.

This result confirms that the entire obstacle detection algorithm discussed in Section 3, is capable of accurately detecting obstacles and effectively representing this obstacle information on a 2D obstacle map, as seen in Figure 14. The algorithm is effective in determining the position of obstacle boundaries relative to the USV, as seen from the camera's perspective. Unfortunately, the entire boundary of an obstacle cannot be mapped, as the camera only has information about the obstacle boundary that faces the camera. However, as the USV moves, it will gain a more complete understanding about the obstacle's boundaries as the cameras see the

obstacles from other directions, enhancing the information present on the obstacle map.

4.3 Results and discussion for outdoor obstacle detection

The obstacle detection algorithm was also tested on a scene captured outdoors. The scene comprises of the ground, a gently inclined grassy slope, and two buoys. The results at the different stages of the obstacle detection pipeline can be seen in Figure 15 on the next page.

When comparing the initial camera images with the final obstacle map, it is apparent that the obstacle detection algorithm has performed effectively in determining the position of obstacles viewed by the cameras. The red and green buoys are cleared isolated and visible in the obstacle map. As discussed in section 3.2.2.2, the BM algorithm parameters min_disparity and disparity_range need to be tuned for the desired detection range. In this case, the desired detection range is approximately 2 – 20 metres, and is achieved by setting the parameters to 0 and 64 respectively. This enables better capturing of depth information about features further from the cameras. Sections of the ground further from the camera are displayed in green and light blue, representing lower disparity values, while sections of the ground closer to the cameras are displayed in shades of red, representing higher disparity values. The red buoy is near the camera and is thus red in the disparity map, while the green buoy is further from the camera and is thus yellow in the disparity map. These results verify that the stereo vision algorithm is correctly determining depth information about the perceived scene.

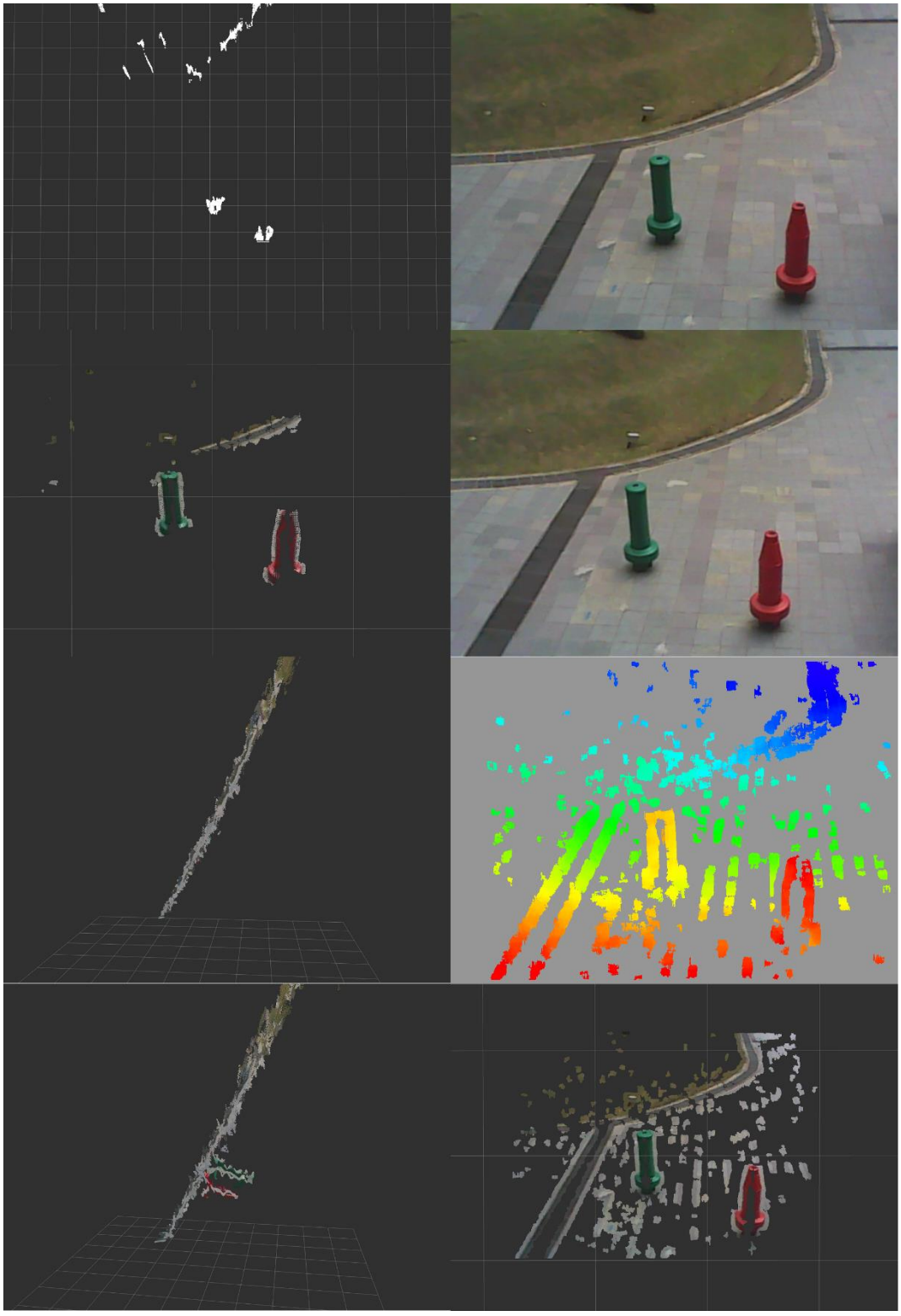

Figure 15: Obstacle detection on an outdoor scene: (clockwise from top right) left camera image; right camera image; disparity map; unprocessed point cloud; point cloud side view; plane segmentation inliers; processed point cloud; obstacle map

The point cloud side view distinctly shows that a large majority of the point cloud lies in the ground plane and can be segmented by the RANSAC algorithm. The algorithm was effective at identifying and removing the points of the point cloud that represent the ground, as seen from the processed point cloud in Figure 15. Sections of the gently inclined grassy slope are also visible in this processed point cloud and in the final obstacle map. This is expected because the grassy slope is not in the same plane as the ground and thus was not extracted by the RANSAC algorithm. More importantly, the two buoys are well represented and defined in the final obstacle map. This end result demonstrates that the obstacle detection algorithm is effective at meeting the objective of the project; detecting obstacles and determining their position relative to the USV in an outdoor environment.

5 Conclusions

The focus of this FYP was to develop a robust obstacle detection algorithm that would enable the USV to autonomously avoid obstacles on the water surface. The proposed approach of combining stereo vision and point cloud processing was developed, improved and implemented to make the USV capable of detecting obstacles in its' environment. The methodology and important algorithms have been discussed and analysed in this report thoroughly. The obstacle detection capability of the algorithm was tested in an indoor and outdoor environment, and the effectiveness of the approach was validated by the accurate obstacle maps generated by the tests. Unfortunately, due to the lack of sea trials, it is difficult to ascertain the effectiveness of the algorithm in the USV's native operational

conditions. Nevertheless, the algorithms have established their reliability in scenarios outside of a sea trial. The obstacle detection algorithm is capable of detecting obstacles and positioning them on an obstacle map relative to the USV for path planning purposes. The favourable results conclude the success of the algorithm.

6 Recommendations for Future Work

Although the obstacle detection algorithm has generated accurate obstacle maps, there are still ways in which the obstacle detection algorithm can be further improved.

6.1 Higher camera resolution and better calibration

One limitation of the current obstacle detection algorithm is that it is incapable of detecting obstacles when the image texture of the obstacle is poor, because the feature matching phase in the BM algorithm will fail. To improve this, a camera of higher resolution could be used to capture the finer details and represent these details in the image that is being processed. Failed feature matching can also be tackled by generating more accurate camera calibration parameters. This can be achieved by using a larger and more dimensionally precise checkerboard for camera calibration. Ensuring that the cameras are fixed in position with respect to one another and using appropriate lighting conditions during camera calibration could also improve the camera parameters that are found.

6.2 Pre-processing and post-processing

Improving the quality of the disparity maps would go a long way in improving the results of the obstacle detection algorithm. This can be done by adding an image pre-processing step which would improve the texture quality of the image before executing the BM algorithm. Better texture quality would mean more successful feature matching and higher quality disparity maps produced by the BM algorithm. A post-processing step can also be applied on the disparity map to smoothen the disparity values and make good estimates for regions with unknown disparity.

6.3 Plane segmentation algorithm

The current plane segmentation algorithm is unable to differentiate between the surface of the water and obstacles that are low and close to the water surface. The plane segmentation approach removes all points that appear to be within the plane of the water surface. However, obstacles that are low and close to the water surface cannot be ignored in some circumstances as they may pose a safety risk to the USV. Thus, it would be worthwhile to investigate other segmentation methods that are capable of differentiating between the water surface and obstacles that lay close to the water surface. One way of achieving this could be by using colour information which is included in the point cloud data, to differentiate between obstacles close to the water surface and the actual water surface.

References

1. Stein, A. D. (2012, January 17). *NPS Acquires Two USVs, Opens Sea Web Lab for Expanded Undersea Warfare Research*. Retrieved from America's Navy: http://www.navy.mil/submit/display.asp?story_id=64803/

2. Kuflex. (2016, August 18). *Overview of StereoLabs ZED camera: lightweight outdoors depth camera*. Retrieved from Kuflex Wordpress: https://kuflex.wordpress.com/2016/08/18/overview-of-stereolabs-zed-camera-lightweight-outdoors-depth-camera/

3. Appiah, N., & Bandaru, N. (2015). *Obstacle detection using stereo vision for self-driving cars.* Palo Alto: Stanford University.

4. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library: Pages 405-458. Sebastopol: O'Reilly. Retrieved from http://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf

5. MathWorks. (2017). *What Is Camera Calibration?* Retrieved from MathWorks Documentation: https://www.mathworks.com/help/vision/ug/camera-calibration.html

6. Bowman, J., & Mihelich, P. (2017, February). *Camera_calibration Package Summary.* Retrieved from ROS.org: http://wiki.ros.org/camera_calibration

7. Simek, K. (2013, August 13). *Dissecting the Camera Matrix, Part 3: The Intrinsic Matrix*. Retrieved from ksimek: http://ksimek.github.io/2013/08/13/intrinsic/

8. McCormick, C. (2014, January 10). *Stereo Vision Tutorial - Part I*. Retrieved from mccormickml: http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/

9. Mihelich, P., Konolige, K., & Leibs, J. (2016, October). *Choosing Good Stereo Parameters*. Retrieved from ROS.org: http://wiki.ros.org/stereo_image_proc/Tutorials/ChoosingGoodStereoParameters

10. PointCloudLibrary. (n.d.). *How to use Random Sample Consensus model*. Retrieved from pointclouds.org: http://pointclouds.org/documentation/tutorials/random_sample_consensus.php#id1

11. Forsyth, D. A., & Ponce, J. (2003). *Computer Vision, a modern approach.* Prentice Hall.

Appendices

Appendix 1: Camera Calibration information

//left camera

image_width: 1024

image_height: 768

camera_name: narrow_stereo/left

camera_matrix:

rows: 3

cols: 3

data: [1451.018154, 0.000000, 549.430767, 0.000000, 1450.109895, 387.926919, 0.000000, 0.000000, 1.000000]

distortion_model: plumb_bob

distortion_coefficients:

rows: 1

cols: 5

data: [0.044613, -0.389761, 0.004343, 0.002730, 0.000000]

rectification_matrix:

rows: 3

cols: 3

data: [0.999073, -0.007567, -0.042377, 0.006880, 0.999843, -0.016345, 0.042494, 0.016039, 0.998968]

projection_matrix:

rows: 3

cols: 4

data: [1492.225034, 0.000000, 615.537750, 0.000000, 0.000000, 1492.225034, 418.899899, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000]

//right camera

image_width: 1024

image_height: 768

camera_name: narrow_stereo/right

camera_matrix:

rows: 3

cols: 3

data: [1446.903083, 0.000000, 528.854146, 0.000000, 1444.002157, 440.583725, 0.000000, 0.000000, 1.000000]

distortion_model: plumb_bob

distortion_coefficients:

rows: 1

cols: 5

data: [0.064075, -0.478818, 0.006540, -0.000218, 0.000000]

rectification_matrix:

rows: 3

cols: 3

data: [0.998491, -0.022663, -0.050017, 0.023470, 0.999603, 0.015615, 0.049643, -0.016765, 0.998626]

projection_matrix:

rows: 3

cols: 4

data: [1492.225034, 0.000000, 615.537750, -481.733076, 0.000000, 1492.225034, 418.899899, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000]