

Online 3D Reconstruction and Ground Segmentation using  
Drone based Long Baseline Stereo Vision System

Prashant Kumar

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in  
partial fulfillment of the requirements for the degree of

Master of Science  
In  
Computer Engineering

Kevin B Kochersberger, Chair  
Ryan K Williams  
Jia-Bin Huang



10<sup>th</sup> September 2018  
Blacksburg, VA

Keywords: Stereo Vision System, 3D Reconstruction, Stereo Calibration, Feature Matching,  
Filtering, Noise Removal

# Online 3D Reconstruction and Ground Segmentation using Drone based Long Baseline Stereo Vision System

Prashant Kumar

## Abstract

This thesis presents online 3D reconstruction and ground segmentation using unmanned aerial vehicle (UAV) based stereo vision. For this purpose, a long baseline stereo vision system has been designed and built. Application of this system is to work as part of an air and ground based multi-robot autonomous terrain surveying project at Unmanned Systems Lab (USL), Virginia Tech, to act as a first responder robotic system in disaster situations. Areas covered by this thesis are design of long baseline stereo vision system, study of stereo vision raw output, techniques to filter out outliers from raw stereo vision output, a 3D reconstruction method and a study to improve running time by controlling the density of point clouds. Presented work makes use of filtering methods and implementations in Point Cloud Library (PCL) and feature matching on graphics processing unit (GPU) using OpenCV with CUDA. Besides 3D reconstruction, the challenge in the project was speed and several steps and ideas are presented to achieve it. Presented 3D reconstruction algorithm uses feature matching in 2D images, converts keypoints to 3D using disparity images, estimates rigid body transformation between matched 3D keypoints and fits point clouds. To correct and control orientation and localization errors, it fits re-projected UAV positions on GPS recorded UAV positions using iterative closest point (ICP) algorithm as the correction step. A new but computationally intensive process of use of superpixel clustering and plane fitting to increase resolution of disparity images to sub-pixel resolution is also presented. Results section provides accuracy of 3D reconstruction results. The presented process is able to generate application acceptable semi-dense 3D reconstruction and ground segmentation at 8-12 frames per second (fps). In 3D reconstruction of an area of size  $25 \times 40 \text{ m}^2$ , with UAV flight altitude of 23 m, average obstacle localization error and average obstacle size/dimension error is found to be of 17 cm and 3 cm, respectively.

# Online 3D Reconstruction and Ground Segmentation using Drone based Long Baseline Stereo Vision System

Prashant Kumar

## General Audience Abstract

This thesis presents near real-time, called online, visual reconstruction in 3-dimensions (3D) using ground facing camera system on an unmanned aerial vehicle. Another result of this thesis is separating ground from obstacles on the ground. To do this the camera system using two cameras, called stereo vision system, with the cameras being positioned comparatively far away from each other at 60 cm was designed as well as an algorithm and software to do the visual 3D reconstruction was developed. Application of this system is to work as part of an air and ground based multi-robot autonomous terrain surveying project at Unmanned Systems Lab, Virginia Tech, to act as a first responder robotic system in disaster situations. Presented work makes use of Point Cloud Library and library functions on graphics processing unit using OpenCV with CUDA, which are popular Computer Vision libraries. Besides 3D reconstruction, the challenge in the project was speed and several steps and ideas are presented to achieve it. Presented 3D reconstruction algorithm is based on feature matching, which is a popular way to mathematically identify unique pixels in an image. Besides using image features in 3D reconstruction, the algorithm also presents a correction step to correct and control orientation and localization errors using iterative closest point algorithm. A new but computationally intensive process to improve resolution of disparity images, which is an output of the developed stereo vision system, from single pixel accuracy to sub-pixel accuracy is also presented. Results section provides accuracy of 3D reconstruction results. The presented process is able to generate application acceptable 3D reconstruction and ground segmentation at 8-12 frames per second. In 3D reconstruction of an area of size 25 x 40 m<sup>2</sup>, with UAV flight altitude of 23 m, average obstacle localization error and average obstacle size/dimension error is found to be of 17 cm and 3 cm, respectively.

## Acknowledgements

First and foremost I sincerely like to thank my advisor Dr. Kevin Kochersberger for believing in me and at every step giving me opportunities as well as helping me out when I got stuck. Computer vision had been a new field for me, which I developed from start during my time at USL. Being provided opportunities to grow and learn new things serves the basic idea of why we join school.

I want to thank my committee member Dr. Ryan Williams for bringing my spirits up and advising me in my desire to change my program to Computer Engineering. Sometimes positive outlook is what one needs and even with a small talk with you one can get ample amount of it.

I want to thank my committee member Dr. Jia-Bin Huang for pursuing teaching the Computer Vision course very passionately and investing a lot to help clear doubts. This course made the basis of my computer vision knowledge and I really liked every minute I worked on it.

I want to thank Dr. John Bird who invested a lot to help guide weekly research and project activities at USL. Even though being busy, you always invested time to check on whether I am on track of my personal goals too besides project goals. This really made the lab feel like home sometimes.

I want to thank my parents, who always stuck with me even during disagreements and specially during hard times. Hard times are what make you mature and really get to know and develop yourself.

I want to thank my friend Arvind and Jitendra, for always being there and specially Arvind for being the mature big brother since the last 10 years.

Time at Virginia Tech was tough for various academic and non-academic reasons. During the course of these two years, I got to learn a lot from life, most times the hard way. Some of the friendships I formed here were like a bright silver lining, some of them I hope continue for lifetime.

I had come to Virginia Tech for one reason which was to become academically proficient in something and I think I do have learned a lot. Besides that during this program I have learned to grow above despair even though falling a lot deeper before coming back. I deeply want to thank this place, the graduate school system and the overall knowledge ecosystem.

## Table of Contents

Abstract.....	ii
General Audience Abstract.....	iii
Acknowledgements.....	iv
Table of Contents.....	v
List of figures.....	viii
List of tables.....	x
Chapter 1: Introduction.....	1
1.1 Introduction.....	1
1.2 Purpose and Motivation.....	2
1.3 Problem Definition.....	2
1.4 Contribution.....	2
1.5 Overview of Thesis.....	3
Chapter 2: Review of Literature.....	4
2.1 Pinhole Camera Model.....	4
2.2 Camera Calibration.....	5
2.3 Stereo Vision.....	6
2.3.1 Stereo Calibration and Rectification.....	7
2.3.2 Semi Global Matching (SGM) Stereo Correspondence Algorithm.....	7
2.4 Feature Matching.....	8
2.5 3D Reconstruction Algorithms.....	9
2.5.1 Structure from Motion.....	9
2.5.2 Bundle Adjustment.....	9
2.5.3 Simultaneous Localization and Mapping.....	9
2.5.4 Visual Odometry.....	10

Chapter 3: System Design .....	11
3.1 Stereo Boom.....	11
3.2 Image Capture Program .....	13
3.3 Stereo Calibration Program.....	14
3.4 Integration with USL’s Multi Robot Coordination & Control System.....	16
3.5 Usage and Experiments.....	17
Chapter 4: Processing Raw Stereo Vision Output.....	19
4.1 Filters.....	21
4.1.1 Statistical Outlier Removal Filter .....	21
4.1.2 Median Filter.....	21
4.1.3 Downsampling using Infinite z-leaf Voxel Grid Filter.....	22
4.2 Clustering Process: Using 2D Texture to produce High Resolution Disparity Images .	24
Chapter 5: 3D Reconstruction .....	27
5.1 Algorithm .....	27
5.2 Algorithm Explanation.....	28
5.3 Correction Step.....	30
5.4 Bad Data.....	32
5.5 Transformation from Image Coordinate to World Coordinates.....	33
5.6 Converting Disparity to Depth: .....	34
5.7 Generating low computation cost semi-dense point clouds by jumping over pixels.....	35
5.8 Ground Segmentation using Sample Consensus Plane Fitting .....	37
Chapter 6: Results.....	39
6.1 Experiment Region.....	39
6.2 3D Reconstruction Results .....	40
6.3 Stereo Vision System Localization and Reconstruction Accuracy .....	43

6.4	3D Reconstruction Speeds .....	43
6.5	Effect of Different Approaches on 3D Reconstruction.....	45
6.6	Ground Segmentation Results.....	48
Chapter 7:	Conclusion and Future Work.....	51
7.1	Conclusion.....	51
7.2	Future Work .....	51
References	.....	52
Appendix A:	Ground truth and 3D Reconstruction results comparison of obstacle localization and size .....	54

## List of figures

Figure 1: Radial distortions in camera lens: Pincushion distortion, No distortion and Barrel distortion .....	5
Figure 2: Depth calculation from stereo images .....	6
Figure 3: Orb feature extraction and all descriptor matches between features of two overlapping images .....	8
Figure 4: Orb features: separating good matches from all possible feature matches between two overlapping pictures .....	9
Figure 5: System design of workflow with hardware and software .....	11
Figure 6: Cameras mounted on stereo rig .....	12
Figure 7: Stereo rig mounted on hexacopter .....	13
Figure 8: Lawn mower path pattern of UAV for image collection.....	17
Figure 9: Picture of data collection experiments being conducted on a setup obstacle course ....	18
Figure 10: Synchronous image capture program .....	14
Figure 11: Flowchart of process to be followed to calibrate the long baseline stereo cameras....	15
Figure 12: Stereo calibration program showing sensor area coverage and rejection of nearby and blurred images.....	16
Figure 13: Rectified left and right camera images post stereo calibration .....	16
Figure 14: Discontinuous stepped output of raw stereo vision.....	20
Figure 15: Specular reflection, white over-exposure and image edge outliers.....	20
Figure 16: Point cloud generated from disparity image without use of Median Filter.....	21
Figure 17: Point cloud generated from disparity image filtered using Median Filter for noise removal with kernel size 31 pixel .....	22
Figure 18: Application of infinite z-leaf Voxel Grid Filter .....	22
Figure 19: Point cloud created with voxel size 0.1 mm; points 105,452,215, file size 1581 Mb.	23
Figure 20: Point cloud created with voxel size 1 cm; points 2,129,850, file size 32 Mb .....	23
Figure 21: Point cloud created with voxel size 2.5 cm; points 343,511, file size 5.2 Mb .....	24
Figure 22: Flowchart of using texture to improve disparity image resolution process .....	24
Figure 23: Process to generate high resolution disparity image .....	25
Figure 24: 2D texture + clustering + plane fitting + noise removal .....	26

Figure 25: Divisions of 3D reconstruction algorithm .....	27
Figure 26: Algorithm employed for online 3D reconstruction .....	28
Figure 27: Feature matched UAV positions, shown in green, fitted over GPS recorded UAV positions, shown in red and blue as sequential rows of Lawn Mower UAV path .....	31
Figure 28: Effect of bad images on sequential feature matched point cloud .....	32
Figure 29: The coordinate transformations from Camera Coordinates to World Coordinates.....	33
Figure 30: Graphically finding angle of misalignment on stereo boom in roll and pitch.....	33
Figure 31: Comparison of Point Clouds generated using jump pixels of (a) 1 pixel (b) 2 pixel (c) 5 pixel and (d) 15 pixel .....	36
Figure 32: Dividing generated 3D reconstruction into smaller blocks .....	37
Figure 33: Segmented ground with Convex hull fitted boundary locations .....	37
Figure 34: Obstacle course setup with 9 obstacles. Image mosaic created using Agisoft PhotoScan software .....	39
Figure 35: 3D reconstruction using Agisoft PhotoScan software.....	40
Figure 36: Dense 3D reconstruction .....	41
Figure 37: Comparison of obstacle ground truth altitudes with 3D reconstruction results .....	41
Figure 38: Localization accuracy visualization .....	42
Figure 39: Comparison of program running speeds with jump_pixels.....	44
Figure 40: Comparison of program running speeds with voxel_size parameter .....	45
Figure 41: 3D reconstruction using Raw GPS-IMU Pose and Raw Stereo Output.....	46
Figure 42: 3D reconstruction using Clustering Process to improve disparity image resolution and noise removal and Raw GPS-IMU Pose .....	47
Figure 43: 3D reconstruction using Clustering Process to improve disparity image resolution and noise removal and Feature Matching Process.....	47
Figure 44: 3D reconstruction using Filtering methods for noise removal and Feature Matching Process .....	48
Figure 45: Comparison of ground segmentation results with different values of jump_pixels for voxel_size 5 cm.....	49
Figure 46: Comparison of ground segmentation results with different values of jump_pixels for voxel_size 10 cm.....	50

## List of tables

Table 1: Lens specifications.....	12
Table 2: Variation in disparity with baseline distance and target depth.....	13
Table 3: Error in depth or depth resolution of stereo vision.....	19
Table 4: Effect of voxel size on point cloud.....	24
Table 5: High resolution disparity image: depth information preservation comparison .....	26
Table 6: Comparison of Feature Matching + ICP based correction step with SLAM.....	31
Table 7: Error in position and altitude of obstacles in 3D reconstruction .....	42
Table 8: Average and standard deviation in localization, altitude and dimension error of obstacles in 3D reconstruction.....	43

# Chapter 1: Introduction

## 1.1 Introduction

Computer vision is a fast evolving field. Last few years has seen a lot of new approaches and new ways of doing things. Machine learning and deep learning have brought in new ways of classification and understanding. LayoutNet[1] presented 3D reconstruction of a room using a deep convolution neural network (CNN) using a single RGB image. Even though there has been a shift in computer vision to use more and more of deep learning, recent years has seen another change which has been in increase in computation speeds of small computers, use of graphics processing unit (GPU) to run parallel computations and also exponential improvement in implementation speeds and capabilities of traditional computer vision algorithms and libraries like stereo vision algorithms, point cloud library (PCL) and OpenCV library. PCL has been one of the most popular library for processing 3D point cloud data. Over the years it has seen its community grow and its implementation capabilities in filtering and removing outliers increase many folds[2]. The speeds and reliability of computations of features has improved by orders of magnitude since the advent of Orb features in 2011[3] as an effective alternative of SIFT and SURF features[4] performance wise. OpenCV has an efficient implementation of matching of features on GPU[5] which includes Orb features. Matching of features on GPU is orders of magnitude faster than on CPU. All this coupled with improvements in hardware, can be extensively exploited to come up with new and improved ways of 3D reconstruction and point cloud filtering methods resulting in more accurate results as well as faster speeds.

This thesis presents design and development of a long baseline stereo vision system, online 3D reconstruction and ground segmentation on a UAV. This project is part of a larger project at Unmanned Systems Lab funded by Industry-University Cooperative Research Centers (IUCRC) Program, aim of which is development of an autonomous Unmanned Aerial Vehicle (UAV), Unmanned Ground Vehicle (UGV) cooperative surveying and analysis system. The 3D reconstruction and ground segmentation would then be used for path planning for UGV.

## 1.2 Purpose and Motivation

During times of natural or man-made disaster, like a hurricane or radiation, there is a risk associated with sending humans as first responders. Examples of these are hurricane Charley in Florida in 2004 and radiation at Chernobyl since 1986. To tackle such situations there is a need of an autonomous air and ground multi robot surveying and analysis system. Such a system can keep humans from harm's way by being the first responders. This thesis work, as small but important part of a larger project at USL, aims to serve and work towards this purpose. This thesis work presents design of a good resolution stereo vision system as well as an efficient and computationally inexpensive algorithm to do 3D reconstruction and ground segmentation. The potential applications of this work extends beyond the mentioned application. Online 3D reconstruction method finds applications in numerous applications like autonomous 3D reconstruction and inspection of buildings or structures, autonomous mapping of places for security applications, mapping of places for use in Artificial Reality (AR), Virtual Reality (VR) applications, autonomous vehicles and robotics, etc.

## 1.3 Problem Definition

The problem is broken down in the following requirements:

- A stereo vision system for low altitude UAV mounted image capture
- Detection of small sized objects using stereo vision
- Light weight system for maximized usage of UAV flight time
- Online 3D reconstruction using camera images, disparity images, GPS and IMU outputs
- Online Ground Segmentation from the 3D point cloud

## 1.4 Contribution

This thesis presents contribution in development of a novel way of online high resolution 3D reconstruction of urban, semi-urban area using stereo vision on an UAV. 3D reconstruction requires high computational power in comparison with computational power available in today's UAV on-board computers. This thesis presents smart ways to come up with semi-dense online 3D reconstruction. The 3D reconstruction process depends entirely on feature matching in 3D. The correction step is used to correct orientation and position of 3D point clouds thus made by fitting

re-projected UAV positions over GPS recorded UAV positions. This is an exponentially fast step compared to localization algorithms based on Sequential Localization and Mapping (SLAM).

The factors of small camera sensor size of 1.3 Megapixel and high altitude or depth of 23 m combine to create resolution of 16 – 20 cm for obstacles at such distance from the stereo camera. Another contribution is in use of 2D texture to improve resolution of disparity images to sub-pixel accuracy, thereby converting raw stepped discontinuous output of stereo vision system to noiseless and continuous 3D reconstruction.

The stereo vision hardware has been built in the lab. A long stereo baseline has been chosen to get higher resolution of obstacles on the ground. Stereo calibration sensitivity to accuracy in planarity of camera mounting increases as the baseline increases. A calibration procedure along with a program has been made to calibrate the long baseline stereo system. The program rejects blurred images as well as nearby images and also provides a graphical output of sensor area coverage.

## 1.5 Overview of Thesis

The thesis has been divided in the following chapters:

1. Introduction
2. Review of Literature
3. System Design
4. Processing Raw Stereo Vision Output
5. 3D Reconstruction
6. Results
7. Conclusion and Future Work
8. Appendix A: Comparing ground truth and 3D reconstruction results of obstacle localizations and sizes

## Chapter 2: Review of Literature

This chapter briefly summarizes the basic ideas on stereo vision, stereo calibration and then explains use of methods used in the presented algorithm like feature matching. Also described here are other popular ways of 3D reconstruction like SLAM and Visual Odometry.

### 2.1 Pinhole Camera Model

To do any sort of conversion of points in an image to points in 3D world, we need a camera model using which this conversion can be done mathematically. The pinhole camera model is a mathematical model of an ideal camera. This ideal camera has aperture of pinhole size, i.e., tending to zero size. This aperture is used to focus light on the camera image frame. This model does not include any lenses or any distortions which come with actual use of lenses like tangential or barrel distortion. Image is formed in the image plane upside down. The distance between the aperture and the image plane becomes the focal length of the camera. A virtual image plane can be assumed to be formed in front of the camera at a distance equivalent to focal length of the camera. Mathematical relation between the 3D points in the world coordinates and the 2D points in the image frame follows:

$$w[x \ y \ 1] = [X \ Y \ Z \ 1]P \quad (1)$$

Where  $X$ ,  $Y$  and  $Z$  represent the coordinates of the point in world frame and  $x$ ,  $y$  the coordinates in the 2D image frame.  $w$  acts as the scaling factor for the image frame and  $P$  is the 4x3 camera matrix which describes the intrinsic and extrinsic camera parameters according to the relation:

$$P = \begin{bmatrix} R \\ t \end{bmatrix} K \quad (2)$$

Where  $R$  and  $t$  are the camera extrinsics, rotation and translation of camera with respect to the world origin in the world frame and  $K$  the intrinsic matrix. The extrinsics matrices transform the world points into the camera coordinates and the intrinsic matrix transforms the camera coordinate points into the 2D image coordinates.

## 2.2 Camera Calibration

Camera calibration needs to be performed to estimate the camera intrinsic and extrinsic matrices. The origin of the camera coordinate system is at the camera's optical center and its  $x$  and  $y$  axis define the image plane.

The intrinsic matrix includes the focal length, optical center and the skew coefficient. The optical center is also called the principal point. The matrix is defined as:

$$K = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \quad (3)$$

where  $s$  is the camera sensor skew,  $f_x$  and  $f_y$  are the focal lengths in the  $x$  and  $y$  directions in pixels and  $c_x$  and  $c_y$  are the camera optical centers in pixels.

Besides the camera intrinsic and extrinsic matrices, lens distortion also needs to be accounted for. These distortions are of two types, radial and tangential. Radial distortion is of two types, pincushion or barrel as shown in Figure 1.

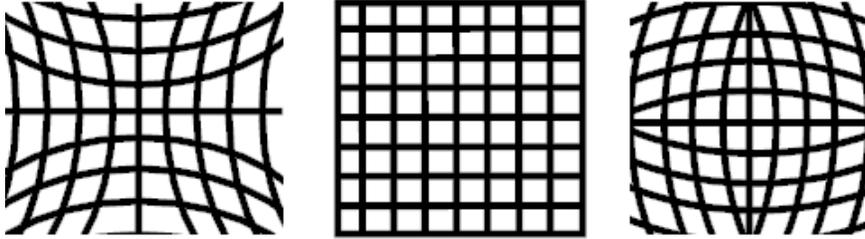


Figure 1: Radial distortions in camera lens: Pincushion distortion, No distortion and Barrel distortion

The radial distortion coefficients model these distortions using the equations[6]:

$$x_{distorted} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \quad (4)$$

$$y_{distorted} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \quad (5)$$

Where  $x$  and  $y$  are the undistorted pixel locations in normalized image coordinates. Normalized image coordinates are calculated by translating pixel coordinates to the optical center and dividing by focal length in pixels making  $x$  and  $y$  dimensionless.  $k_1$ ,  $k_2$  and  $k_3$  are the radial distortion coefficients.  $r^2$  is  $x^2 + y^2$ . Typically, first two coefficients are sufficient for calibration.

Second type of distortion is tangential distortion. This occurs if the lens and image plane are not parallel. This is modelled using the following equations:

$$x_{distorted} = x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)] \quad (6)$$

$$y_{distorted} = y + [2 * p_2 * x * y + p_1 * (r^2 + 2 * y^2)] \quad (7)$$

Here  $p_1$  and  $p_2$  are the tangential distortion coefficients of the lens and  $r^2$  is again  $x^2 + y^2$ .

### 2.3 Stereo Vision

Stereo vision estimates depth of an object of interest using two pictures taken from two cameras at the same time. These cameras need to be aligned together in one plane and in one horizontal line. Assuming this is the case, graphically the system configuration can be represented[7] as shown in Figure 2.

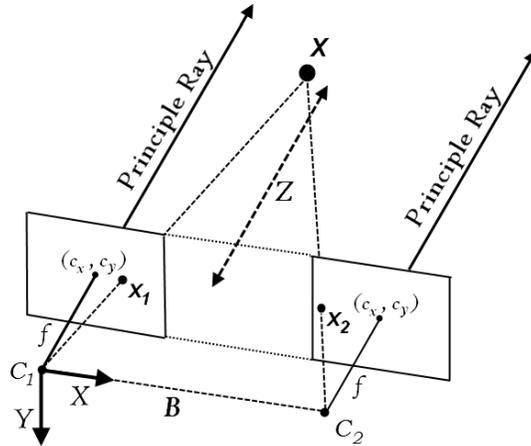


Figure 2: Depth calculation from stereo images

In this configuration the distance  $Z$  from the 3D point  $X$  to the stereo system can be found by the following equation:

$$Z = \frac{fB}{x_1 - x_2} = \frac{fB}{d} \quad (8)$$

where  $B$  is the stereo baseline, i.e., the distance between the left and right cameras,  $f$  is the focal length in pixels of the lenses,  $x_1$  and  $x_2$  are the  $x$  coordinates of the point  $X$  in the first and second images, respectively.

### 2.3.1 Stereo Calibration and Rectification

As mentioned in the previous section, to estimate depth both the left and right cameras need to be in the same horizontal line as well as in the same plane. As with any hardware, this is not the case and planarity as well as being on the same line is to a level of accuracy. To make both the camera images in the same plane and horizontal line, rotation and translation between the cameras need to be estimated. Stereo calibration does this. Using this rotation and translation either of the image can be transformed to align well with the other image. This process is called stereo rectification.

### 2.3.2 Semi Global Matching (SGM) Stereo Correspondence Algorithm

SGM, presented by Hirschmuller[8], uses a mutual information based pixel matching cost and aggregates matching costs in multiple directions. Hirschmuller provides a detailed description. The energy function that needs to be minimized is:

$$E(d) = \sum_p (C(p, D_p)) + \sum_{q \in N_p} P_1 T[|D_p - D_q| = 1] + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1] \quad (9)$$

where  $D_p$  is the pixel disparity of pixel  $p$ ,  $q$  is neighbor of  $p$ ,  $C$  is the cost function and  $N_p$  is neighborhood of point  $p$ , in all directions. The function  $T$  returns the weight of penalty costs  $P_1$  and  $P_2$ . The combination of SGM with different kinds of local similarity metrics has been found to be insensitive to noise and changes in lighting. This makes SGM efficiently deal with large untextured areas[9].

Kalarot et al. makes a performance comparison of various stereo vision algorithms running on GPU[10]. The algorithms compared are Symmetric Dynamic Programming Stereo (SDPS), SGM, Block Matching (BM), Belief Propagation (BP), Belief Propagation – Constant Space (BP\_CS) and Semi Global Block Matching (SGBM). The performance comparison is done on speed and throughput, scalability and accuracy. His experiment results found SGM to have good accuracy as well as speed.

D. Hernandez-Juarez et al. presented his work of embedded real-time stereo estimation via SGM on the GPU[9]. His implementation on NVIDIA Tegra TX1 computer was presented to provide real-time stereo results at 42 fps for rectified stereo images of resolution 640x480.

## 2.4 Feature Matching

In images, a feature is a numerical representation of uniqueness of a pixel or a set of pixels. This being a numerical value, can be used to perform mathematical operations like matching similar pixels. Over the years many types of features have been invented, popular being Scale-invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Binary Robust Independent Elementary Features (BRIEF), Oriented FAST and rotated BRIEF (Orb). Orb is a fusion of FAST and BRIEF descriptors with some modifications. Orb features are two orders of magnitude faster than SIFT and one order of magnitude faster than SURF features, while still performing equivalently or better than them in feature detection[3][4].

The result of feature detection is a keypoint with a location and descriptor value. Similar features have nearby values of their descriptors. Feature matching can be performed by finding the nearest descriptor value. This can be done by finding the Euclidian distance between descriptors. For any descriptor, there would be multiple matches with increasing distance between them as shown by Figure 3.

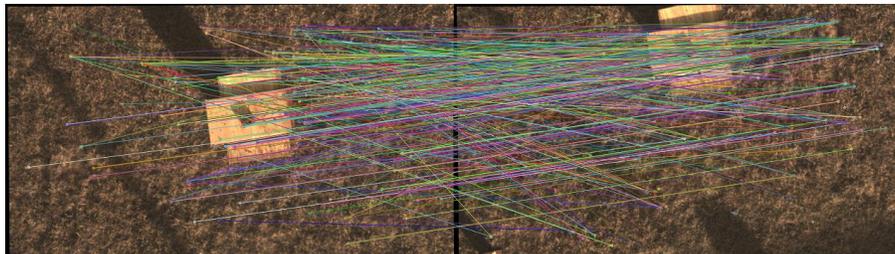


Figure 3: Orb feature extraction and all descriptor matches between features of two overlapping images. As seen, many descriptors have incorrect false matches that need to be rejected.

As can be seen, out of all matches of a feature, the best match needs to be found. This can be found by putting a ratio threshold between the best and second match as well as a maximum distance value for the best match. Applying these, the good matches of descriptors can be found as showing in Figure 4. These good match correspondence can be used for estimating camera motion between the two overlapped images.

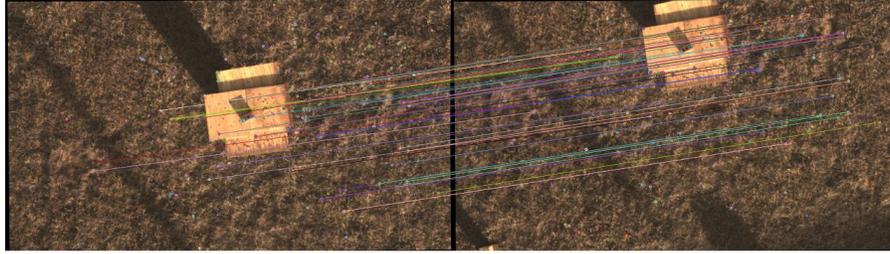


Figure 4: Orb features: separating good matches from all possible feature matches between two overlapping pictures

## 2.5 3D Reconstruction Algorithms

3D reconstruction in computer vision is the process of capturing the shape and appearance of real objects. This is different from an image since an image is in 2 dimensions while 3D reconstruction is in 3 dimensions, similar to real world objects.

### 2.5.1 Structure from Motion

Structure from motion works on point correspondence between sequence of images. Here keypoints need to be tracked using algorithms like Kanade-Lucas-Tomasi (KLT) algorithm. Post tracking of keypoints, with use of camera fundamental matrix and triangulation, the relative pose of consecutive images relative to previous images can be found, creating 3D locations for keypoints as well as camera poses.

### 2.5.2 Bundle Adjustment

Bundle adjustment (BA) is the process of simultaneously optimizing both 3D point locations and camera pose for visual reconstruction[7], [11]. It does this by basically minimizing the re-projection errors between image locations of observed and predicted image points. The result of BA is a refined more fitted 3D reconstruction.

### 2.5.3 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a computational problem of constructing and updating a 3D map captured from a moving agent while simultaneously keeping track of the agent's location within the 3D map[12]. There are multiple implementations of SLAM and it is

generally tailored for required applications. SLAM works on keeping a state of every entity it is keeping track of and with added information improving the respective states by finding the combined minimum error solution. Benefits of SLAM include continuous improvement in state estimate with added data, but simultaneously increase in computations with increased map size.

#### 2.5.4 Visual Odometry

Visual Odometry is the method of determining the position and orientation of a robot or vehicle by analyzing correspondence between sequential images. This uses feature matching to a large extent. Visual odometry is significantly faster compared to SLAM but suffers from incremental errors.

## Chapter 3: System Design

This chapter details the hardware and software of the system. Figure 5 graphically shows the design of workflow of the system along with hardware and software. In the following sections, each block is described in more detail.

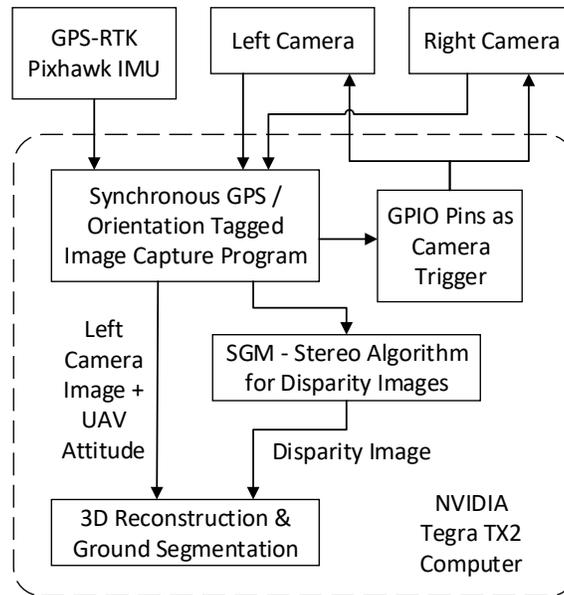


Figure 5: System design of workflow with hardware and software

### 3.1 Stereo Boom

The custom built stereo boom consists of two cameras mounted on a 1" carbon fiber tube. The carbon fiber tube is selected for its durability and rigidity, which is highly important in stereo calibration. The mounts of the cameras have been 3D printed. These 3D printed mounts have compliance in them, which is made use of during adjustments to calibrate the stereo well. The cameras chosen are E-con systems See3CAM\_10CUG cameras. These are 1.3 Megapixel (MP), global shutter cameras with external trigger functionality. These run using USB ports. The lens chosen are 16 mm focal length lenses with 5 MP quality and calibrated and marked manual focus and manual aperture. The stereo boom is shown in Figure 6. Shown are 3 cameras on the rig as initially different baselines were planned to be used. The final baseline on the system for experiments is 60 cm. This is a long value for a baseline. Commercial stereo cameras available in

the market have baseline far lower under 10 cm. A benefit of long baseline is better resolution at higher altitudes or depths. More depth resolution using stereo cameras has been discussed in Chapter 4:. Aperture size controls the amount of light coming in. It also indirectly controls the depth of field. We chose a small aperture size, giving us a good depth of field in range of 10 m to 40 m. The angle of view and focal length of the lenses is mentioned in Table 1.

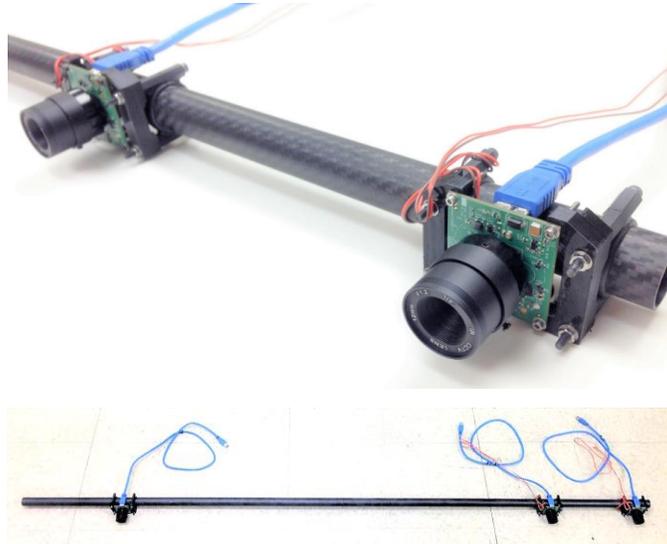


Figure 6: Cameras mounted on stereo rig

Table 1: Lens specifications

F	16	mm
f (in pixel)	266.67	pixel
angle of view	22.08	degree
	16.56	degree

Table 2 lists the expected disparity values with a stereo baseline of 60 cm at different depths. The stereo algorithm implementation chosen has a max disparity value of 128. Therefore, with this taken into account the flight operation range of the system becomes 20 m - 40 m.

Table 2: Variation in disparity with baseline distance and target depth

Depth $z$ (m)	Disparity $d$ (pixels, $d=fb/z$ )
10	256.0
12	213.3
20	128.0
23	111.3

Figure 7 shows a picture of stereo boom mounted on UAV. At time of this picture being taken the stereo baseline was higher at 1.5 m. The on-board computer is NVIDIA Tegra TX 2. The UAV has GPS using GPS-RTK system and its controller is Pixhawk controller with an IMU.



Figure 7: Stereo rig mounted on hexacopter

### 3.2 Image Capture Program

A program in C++ was made to note GPS and IMU attitude from Pixhawk using MAVLink and trigger and capture images. This trigger functionality was made using the GPIO pins on Tegra TX2 computer. Using these GPIO pins the cameras are triggered using their external trigger circuits. To fetch images from the cameras, UVC Library was used, which is a library to use video devices. Salient features of the program are:

- Program in C++
- Synchronized image capture from 3 1.3MP cameras at 5fps, program uses multi-threading using boost for image conversion from Raw Bayer format to RGB and saving to disk
- MAVLink GPS + IMU data logging in text comma separated format
- Stress testing – 32mins no-crash continuous run captured 9506 (x3) 1280x960px (3.5Mb) pictures (~100Gb data) at an average of 4.89Hz along with data logging
- Stores experiment data in new folder with timestamp, save program log
- Uses a settings.xml file to load modifiable runtime settings and variable values
- Live visualization
- Can set brightness and exposure value to camera at program start

Figure 8 shows screen grab of the program with visualization of three cameras during runtime.

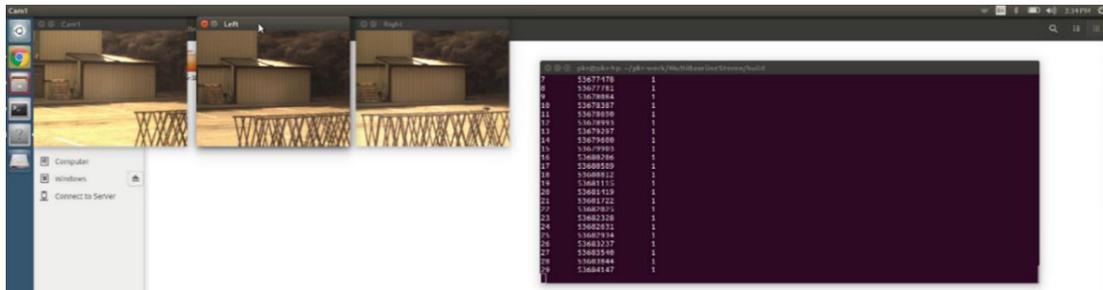


Figure 8: Synchronous image capture program

### 3.3 Stereo Calibration Program

A stereo calibration program has been developed based on the open source stereo calibration program of OpenCV. Three salient features of this program are:

- Method to reject blurred images based on a threshold value by calculating blurriness in located checker box on the image
- Method to reject nearby images based on a threshold value by calculating pixel location of checker box on the image
- Live visual feed of sensor area coverage on both cameras to user

Besides this calibration program, a process to calibrate the long baseline stereo has been developed. It has been observed that it is considerably difficult to calibrate stereo system as the baseline and focal length of lens increases. With these increases the location of image plane becomes more sensitive to orientation changes and the OpenCV calibrator program finds it difficult to estimate the stereo calibration parameters. The process is described in the flowchart shown in Figure 9.

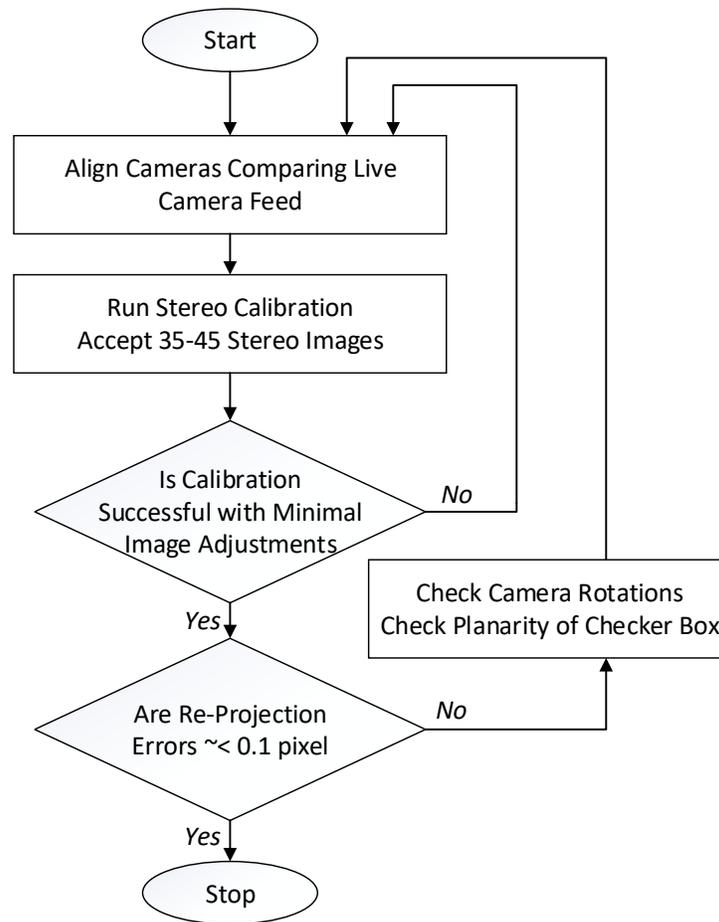


Figure 9: Flowchart of process to be followed to calibrate the long baseline stereo cameras

It has been experimentally found that 35-45 stereo images produce the best calibration results. Employing the above shown process results in re-projection errors of under 0.1 pixel. Figure 10 shows a screen grab of the stereo calibration program in action while Figure 11 shows a rectified image pair.

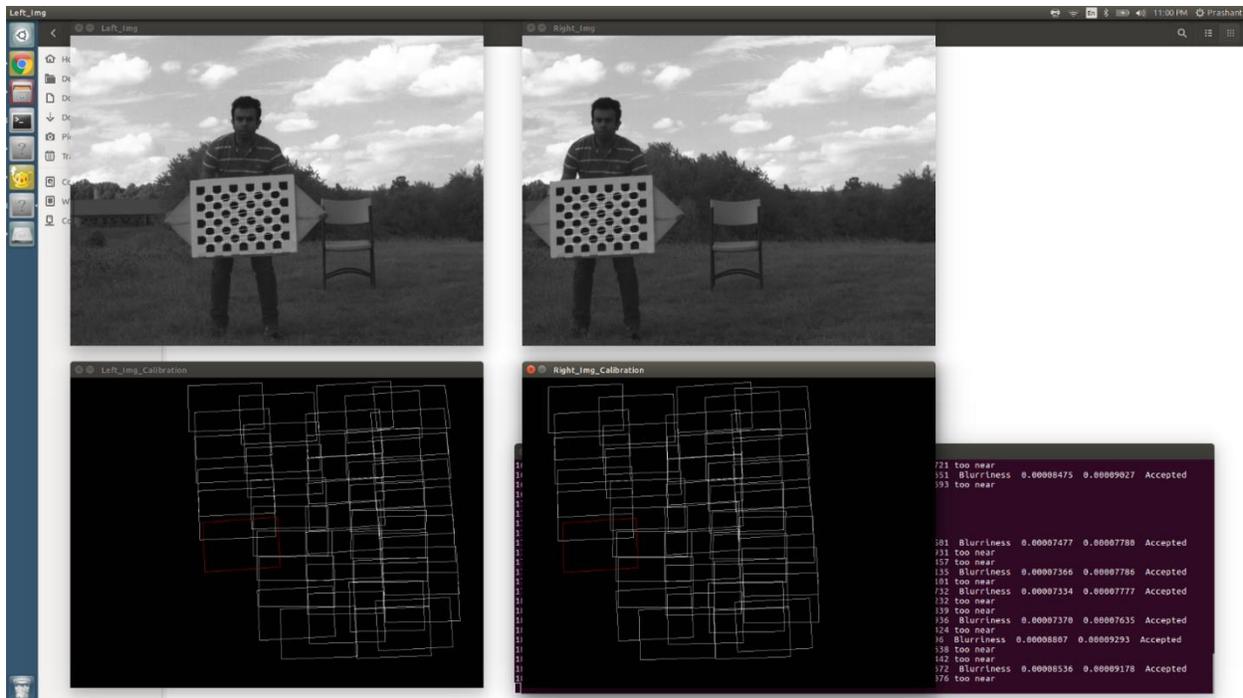


Figure 10: Stereo calibration program showing sensor area coverage and rejection of nearby and blurred images

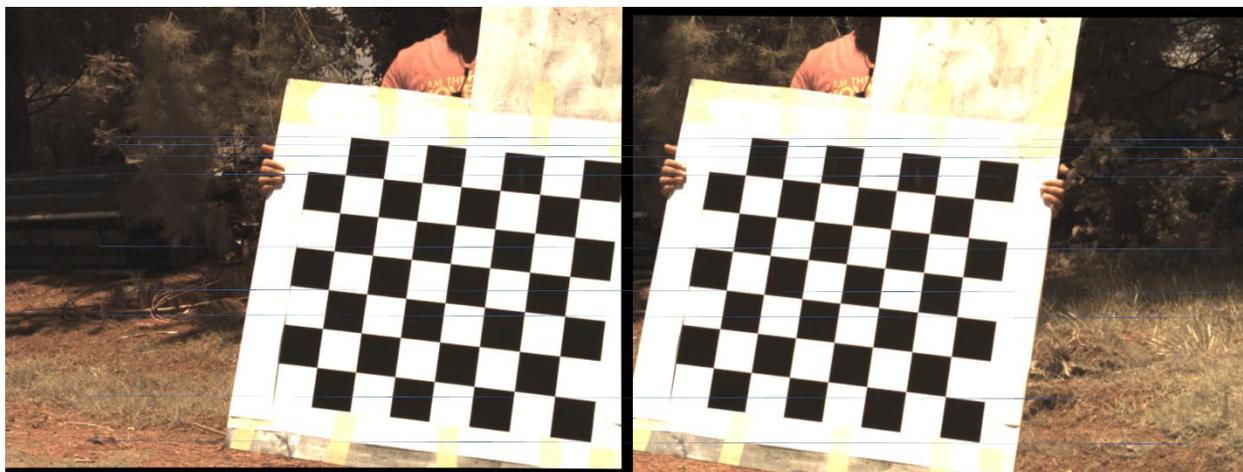


Figure 11: Rectified left and right camera images post stereo calibration

### 3.4 Integration with USL's Multi Robot Coordination & Control System

Over the years USL has developed a ROS based multi-robot communication and control system. This system communicated using wifi. Using this stand-alone image capturing program as base

and modifying its function structure, the program was integrated to the ROS system with the help of lab members. This made the ROS system have images in live feed available over wifi as well as to any publisher on the system. The SGM stereo program was also made a ROS node to publish disparity images in real time.

### 3.5 Usage and Experiments

UAV data collection experiments were designed with the idea of implementing 3D reconstruction using feature matching. Good feature matching requires lot of overlap between images. Based on these requirements, lawn mower path pattern for UAV was chosen. The speed of the flights was kept as 2 m/s. At these flights image was captured at 5 fps. The UAV was flown at 23 m height. This gave a sequential image overlap of 93%. The row distance in lawn mower pattern was kept as 1.5 m. This gave a side overlap of 75%. This path pattern is shown in Figure 12.



Figure 12: Lawn mower path pattern of UAV for image collection

Figure 13 shows a picture of data collection experiments being conducted on a setup obstacle course. The UAV can be seen in air along with the stereo boom mounted on it.



Figure 13: Picture of data collection experiments being conducted on a setup obstacle course

## Chapter 4: Processing Raw Stereo Vision Output

Stereo vision, as discussed in Section 2.3, generates a disparity image. This disparity image has pixels in integer values between 0 to 255 for a uint8 image. Since these are integer values, using these disparity values will have a resolution for different depths. We know that depth can be calculated using the equation

$$Z = \frac{fB}{x_1 - x_2} = fBd \quad (8)$$

Because of integer values of disparity, the best resolution we can get is of 1 pixel disparity. Therefore, the resolution in depth can be calculated using the equation:

$$\begin{aligned} \Delta Z &= \frac{fb}{d-1} - \frac{fb}{d} \\ \Rightarrow \Delta z &= fb / \{d(d-1)\} \end{aligned} \quad (10)$$

Substituting  $d$  we get:

$$\Rightarrow \Delta z = z^2 / (fb - z) \quad (11)$$

Using the above formula, we can calculate the depth resolution for different depths based on the 60 cm baseline and 1.3 MP cameras to be:

Table 3: Error in depth or depth resolution of stereo vision

Depth $z$ (m)	Resolution Error in Depth $\Delta z$ (cm)
10	3.9
12	5.7
20	15.7
23	20.9

From the above table we see that at altitudes of 23 m, the best depth resolution we can get is of 20.9 cm. The effect of this is that generating 3D point clouds using raw stereo vision output will give discontinuous steps in the result. This is shown in Figure 14.

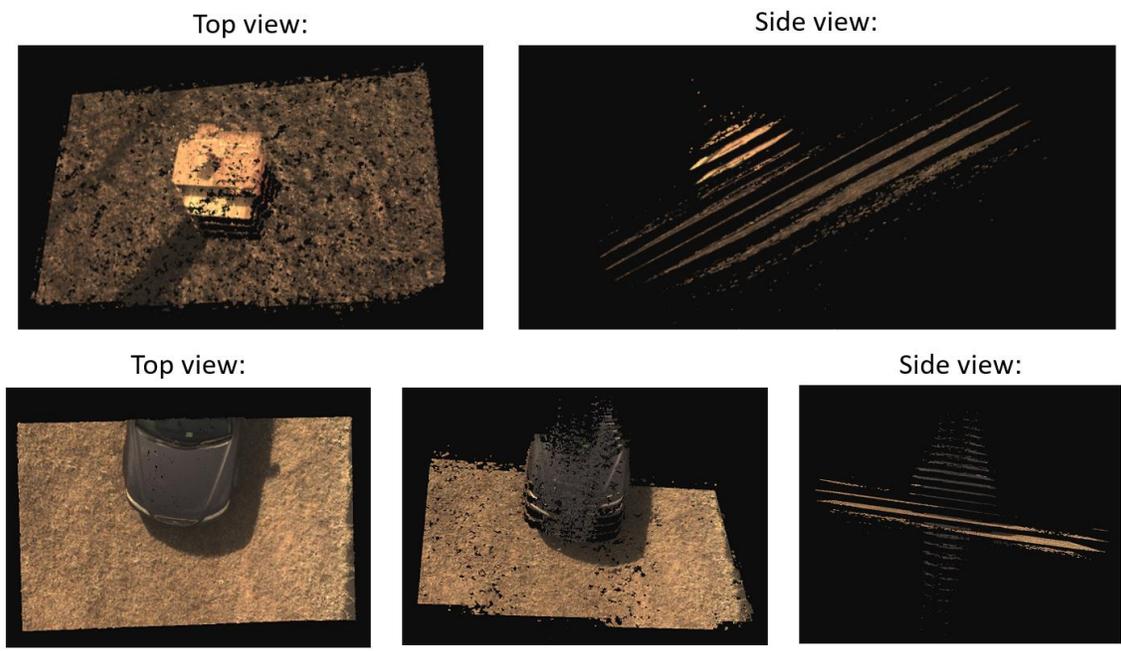


Figure 14: Discontinuous stepped output of raw stereo vision

Therefore, to make meaningful use of such discontinuous stepped stereo output, it needs to be processed a lot. Besides this issue, a big issue faced by stereo output is of outliers. We observed three types of outliers:

- Specular reflection outliers
- White over exposure outliers
- Outliers at image edges

The above mentioned outliers are shown in Figure 15.

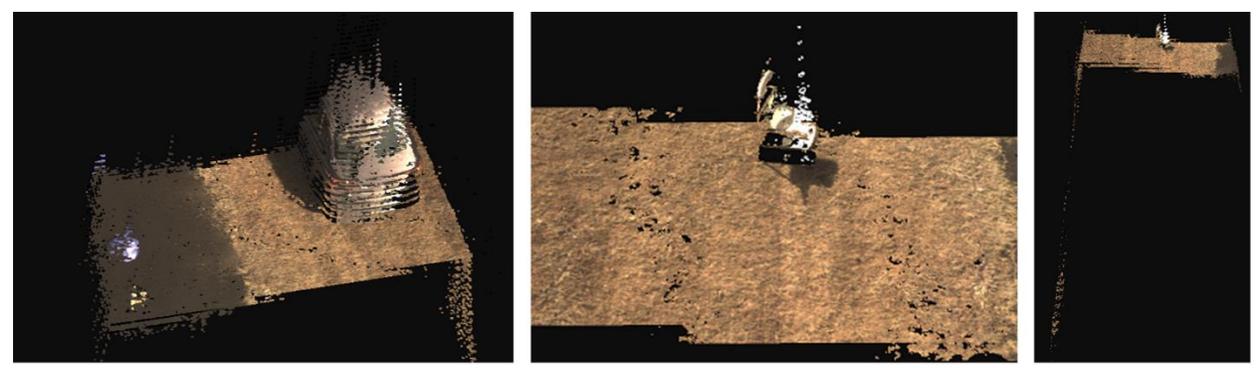


Figure 15: Specular reflection, white over-exposure and image edge outliers.

## 4.1 Filters

To correct various issues in raw stereo output, and in general in improvement of a 3D point cloud, filters can be employed. The employed filters along with their effect is described here. All these filters have been used from the PCL library.

### 4.1.1 Statistical Outlier Removal Filter

The statistical outlier removal algorithm iterates over an input twice. During the first iteration the average distance for each point is computed from its nearest  $k$  neighbors. Then mean and standard deviation of these distances is computed to determine a distance threshold. This threshold is equal to  $\text{mean} + \text{stddev\_multiplier} * \text{stddev}$ . In the next iteration points will be classified as inliers or outliers based on their distance being above or below this threshold value. We observe that this filter has a major effect in removal of outliers.

### 4.1.2 Median Filter

Median filter finds the median value in a block of pixels. This is a very useful filter to remove outliers as generally outliers have sparse presence compared to other pixels. Using median filter noise from the disparity image can be reduced to generate low noise point cloud as shown in Figure 17 compared with no filter cloud in Figure 16.

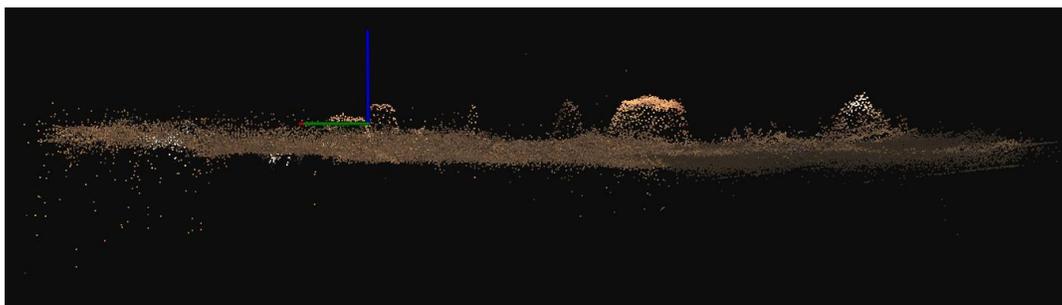


Figure 16: Point cloud generated from disparity image without use of Median Filter

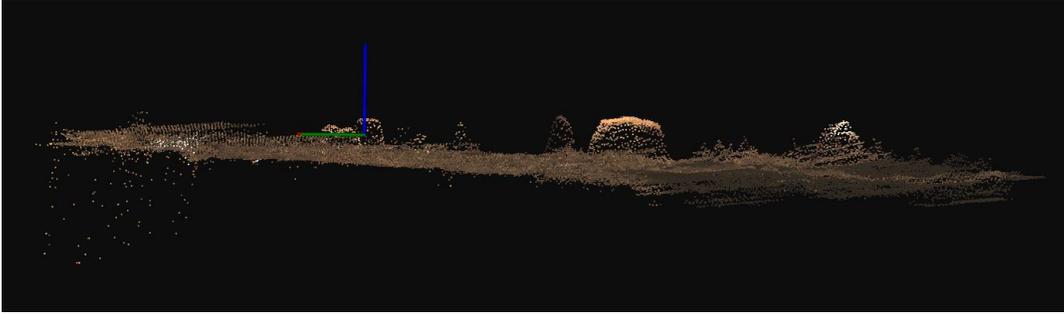


Figure 17: Point cloud generated from disparity image filtered using Median Filter for noise removal with kernel size 31 pixel

Though it shows good results, in sparse point clouds the application of median filter is difficult as the number of points is less.

#### 4.1.3 Downsampling using Infinite z-leaf Voxel Grid Filter

Overlaying multiple single image point clouds on each other gives many z values for every x and y values. A good way to combine the information is to find centroid of all z points in a cuboidal voxel of equal x and y size but infinite z size. A before and after picture of application of this filter is shown in Figure 18.

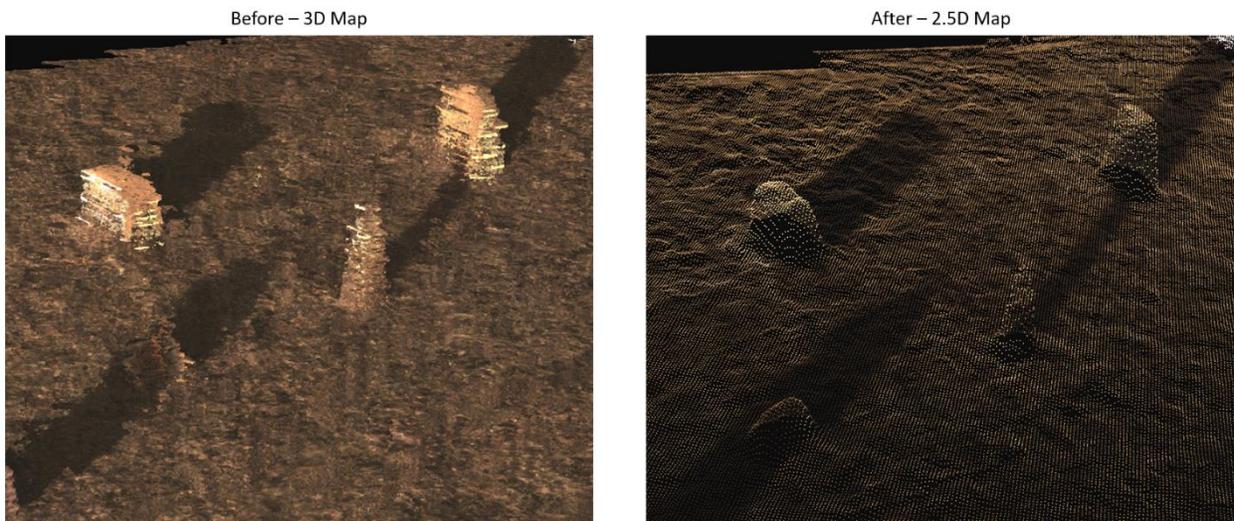


Figure 18: Application of infinite z-leaf Voxel Grid Filter

Besides converting the map into a 2.5D Digital Elevation Map (DEM), another effect of this filter is in reduction of number of points as well as reduction of file size. This is shown in Figure 19, Figure 20, Figure 21 and Table 4.

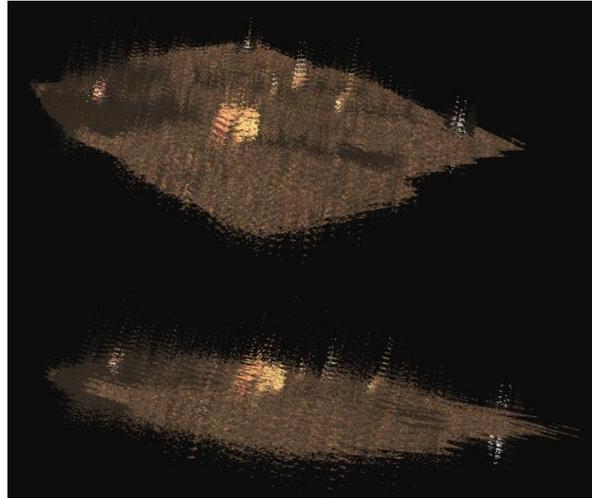


Figure 19: Point cloud created with voxel size 0.1 mm; points 105,452,215, file size 1581 Mb

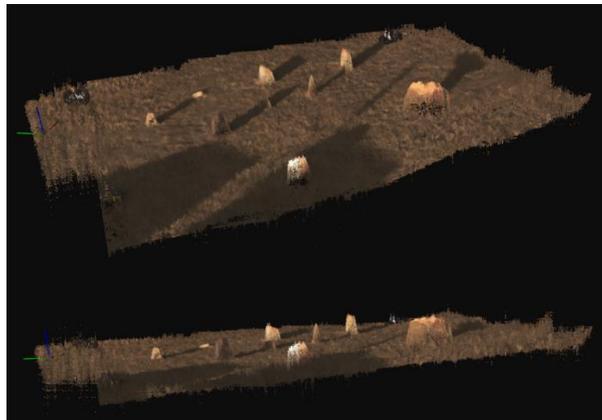


Figure 20: Point cloud created with voxel size 1 cm; points 2,129,850, file size 32 Mb

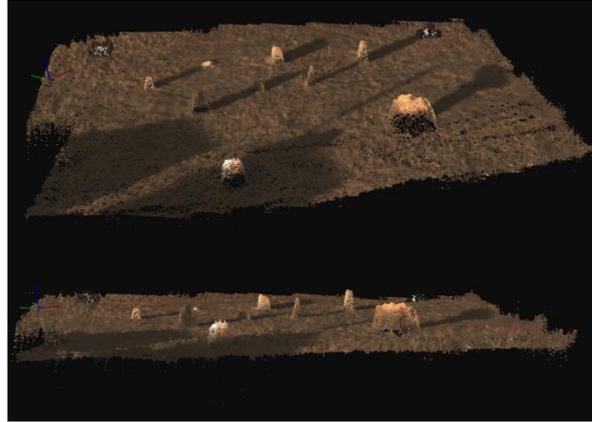


Figure 21: Point cloud created with voxel size 2.5 cm; points 343,511, file size 5.2 Mb

Table 4: Effect of voxel size on point cloud

voxel_size (cm)	Number of Points	File Size (Mb)
0.01	105,452,215	1,581
0.1	68,298,921	1,024
0.5	8,323,425	125
1	2,129,850	32
2.5	343,511	5.2
5	86,462	1.3

#### 4.2 Clustering Process: Using 2D Texture to produce High Resolution Disparity Images

A new process is presented here to use texture information in an image to improve resolution of disparity image. Benefit of this process is that it reduces noise and outliers and provides sub-pixel resolution to disparity image. Point clouds generated from these disparity images have more continuous surfaces are most locations are more near to ground truth than discontinuous point clouds. Figure 22 describes the process in a flowchart while Figure 23 shows the process graphically.



Figure 22: Flowchart of using texture to improve disparity image resolution process

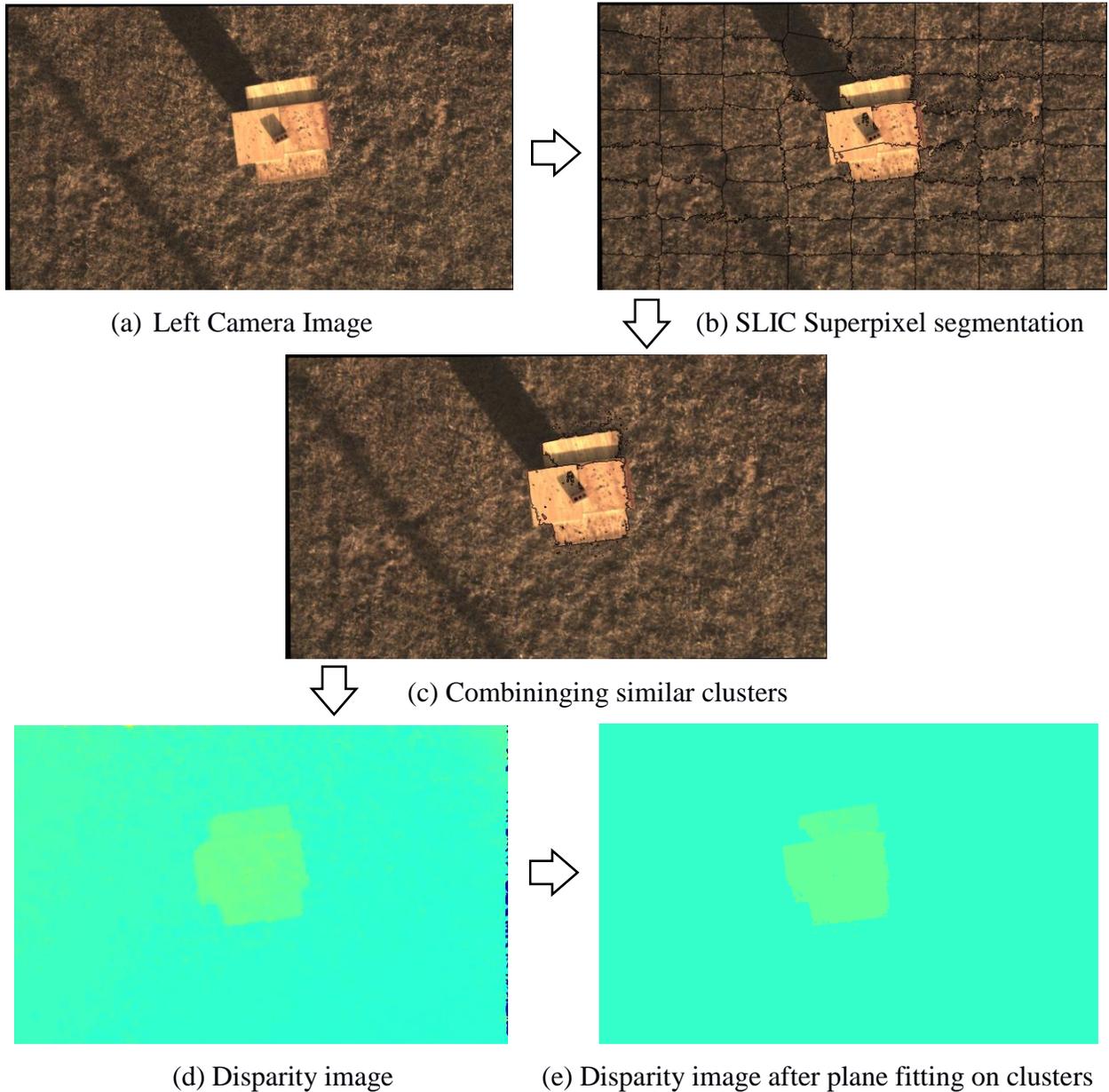


Figure 23: Process to generate high resolution disparity image

The left camera image is segmented using SLIC Superpixels[13] on an assumption that similar texture will assign a to a similar class. Similar clusters are joined based on a distance function which includes ‘Lab colors’ as well as variance in Lab colors in a cluster. This process essentially increases the resolution of the disparity image by giving it sub-pixel resolution. Using variance in Lab colors in the distance function makes sure the clusters contain similar information including

in terms of shadow, color and luminosity and thus dis-similar clusters with different shape or structures are not joined together. Using this a label map is generated which contains information on cluster label for each pixel. Using this label map, a plane is fitted over disparity values in each cluster.

Figure 24 shows a point cloud generated using high resolution disparity image thus made. Table 5 shows comparison of preservation of depth information using this process.

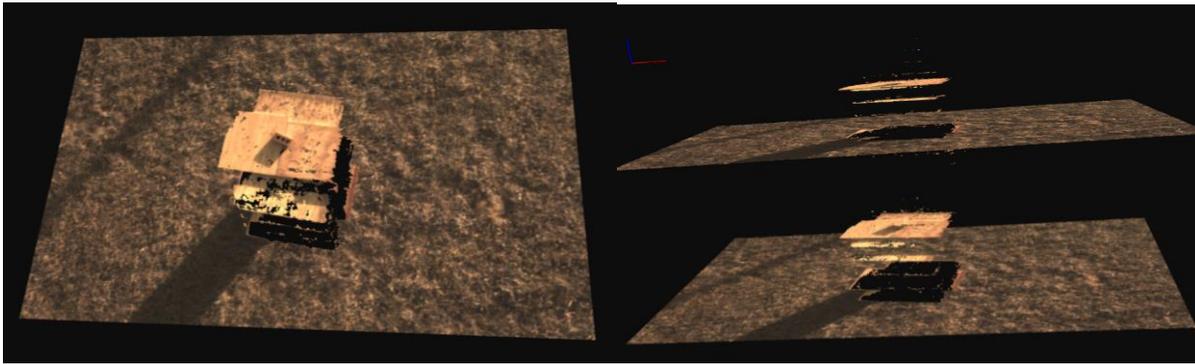


Figure 24: 2D texture + clustering + plane fitting + noise removal

Table 5: High resolution disparity image: depth information preservation comparison

Location	Original 3D image depth (m)	Improved 3D image depth (m)	Difference %
Top left corner	23.53	23.49	0.17%
Bottom left corner	23.72	23.71	0.04%
Bottom right corner	24.20	24.19	0.04%
Top right corner	23.94	23.93	0.04%
Center Table	22.90	22.86	0.17%

The benefit of this clustering process is that it removes or averages out outliers from the single image point cloud, thereby enabling us to visually see how exactly the 3D reconstruction process is working. One major drawback of this process is the computation cost. To process an image it takes around 30 seconds to complete. Because of this drawback this process was not used in the final 3D reconstruction process.

## Chapter 5: 3D Reconstruction

This section describes the 3D reconstruction method employed using multiple stereo image pairs and UAV positions and orientations. The algorithm has been described in the following section. After understanding how to process raw stereo vision results and clean outliers, the challenge at every step had been to do it at a faster speed. The whole algorithm and every step has been designed keeping this in mind.

### 5.1 Algorithm

The algorithm is divided into three blocks: Feature Matching and 3D Fitting, Orientation Correction and 3D Reconstruction and 2.5D DEM and Ground Segmentation as shown in Figure 25. New images keep flowing into the algorithm in the first block. The first block keeps looping in itself until new images coming in are from the same row of the lawn mower pattern. When the row shifts, the control moves to orientation correction block. Orientation is corrected here and new image point clouds added to the point cloud. Post this the thread again goes back to the first block. The third block runs on a parallel thread to keep the program running. The third block is downsampling block and is one of the slowest steps. Employing this algorithm we are able to do 3D reconstruction online. Figure 26 shows the algorithm steps in detail.

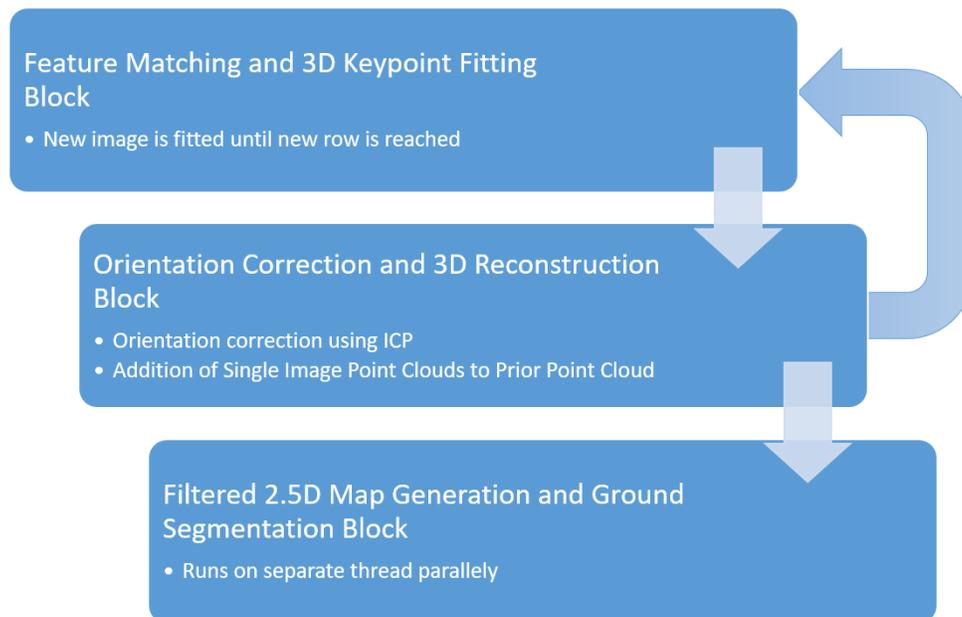


Figure 25: Divisions of 3D reconstruction algorithm

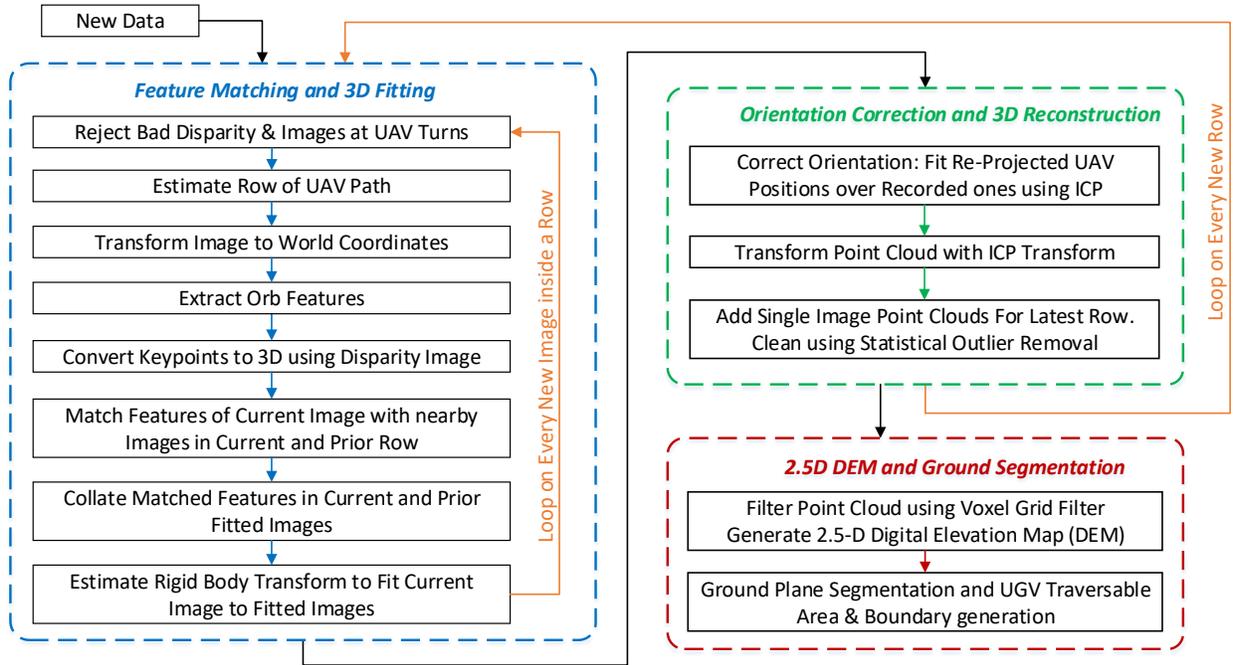


Figure 26: Algorithm employed for online 3D reconstruction

## 5.2 Algorithm Explanation

For 3D reconstruction, the following process has been followed:

### *Block 1: Feature Matching and 3D Keypoint Fitting Block*

- Reject bad disparity images caused by hardware issues by doing variance calculation. This has been described in Section 5.4.
- Estimate if new image coming in is in the current row or new row. This is done by employing RANSAC line fitting over consecutive UAV GPS positions. If new UAV position is away from a threshold value then the row has ended
- Reject images while UAV is taking a turn. UAV follows a lawn mower pattern during image capturing. When the UAV is taking a turn its orientation changes quite a bit. Highly angled images are rejected as these have no or less overlap and it becomes difficult to fit them correctly. This is checked by the fitted row. Locations outside the fitted row are rejected.
- Find the transformation required to transform image from image coordinates to world coordinates using the GPS Pixhawk-IMU estimated UAV pose as mentioned in Section 5.5.

- Find Orb features in left camera images.
- Convert 2D matched keypoints to 3D using disparity image as mentioned in Section 5.6. Transform these 3D points to world coordinates by the transform calculated 2 steps back.
- Match the Orb features between images with UAV positions near a threshold distance from the new image UAV position. Images considered for this matching are in current and the previous row. This results in good fitting. To speed up the process, feature matching function of OpenCV with CUDA which utilizes GPU processing is used.
- Find rigid body transformation between new image keypoints and previously fitted keypoints. Also apply the feature matched transformation to image UAV position and record it.

*Block 2: Orientation Correction and 3D Reconstruction Block*

- Just feature matching fitting keeps building up on the orientation error of the first image. This produces a dis-oriented but fitted point cloud. A correction step is needed to correct its orientation. A computationally in-expensive way to correct its orientation is to fit re-projected UAV positions of feature-matched point cloud to GPS recorded UAV positions using Iterative Closest Point (ICP) algorithm. This correction step is explained in detail in Section 5.3.
- Apply the ICP calculated correction transformation to the fitted point cloud to correct its orientation.
- The above process calculates the transformations required for fitting each image point cloud together. Since number of keypoints in an images comes out to be far lower at around 1500 points than actual image points which is 1.3MP, the above calculations are computationally inexpensive.
- Post calculation of required transformations, point clouds for each image are generated and transformed using the above calculated transformation and added to a combined point cloud. To speed up the process, multi-threading using Boost library is used to generate individual image point clouds to utilize all computing power. At this step to speed up the process even more fewer image pixels can be picked instead of all pixels. This can be done by jumping over pixels in x and y directions of the image. This jump\_pixel process has been described in Section 5.7.

### *Block 3: Filtered 2.5D Map Generation and Ground Segmentation:*

- This step is computationally expensive because of downsampling and hence is run in a parallel thread.
- Downsample or filter the combined point cloud to voxels of equivalent x y direction leaf size and infinite z size. Find centroid of all the points inside these voxels to convert the point cloud to a 2.5d map. Now only one point in the z direction would come as a result in each voxel. Cameras from above also pick only the highest point in the z direction of the terrain. This filtering averages out inaccuracies in the point cloud and also reduces its file size.
- Run Ground Segmentation to segment out ground from the fitted point cloud. This can be used by UGV as traversable obstacle free region on the ground and various paths can be planned on it. This segmentation process has been described in Section 5.8.

### 5.3 Correction Step

The correction step implemented in the algorithm is an innovative and novel way to correct orientation of the feature matched point cloud in a computationally inexpensive way. The GPS positions have local error but several GPS positions combined together averages the error to a lower value. When several images have arrived, a good way to correct orientation of the feature matched point cloud is to match the feature matched point cloud with GPS-IMU pose point cloud. Because of sensor error, there would be local error in the GPS-IMU pose point cloud but since there are no incremental errors in GPS position, there is low global localization error in this point cloud. Therefore, by fitting the feature matched point cloud over the GPS-IMU pose point cloud, we can get the best of both worlds – good local localization from feature matched point cloud and good global localization from GPS-IMU pose point cloud. To do this Iterative Closest Point (ICP) algorithm can be used. With the added benefit of this step, the drawback of this step is computational expensiveness. Since the number of points in the 3D reconstructed point cloud is in orders of millions of points, the ICP transformation takes time. Delving deeper in the solution, we see that we don't actually require all the points in the point cloud. We here only require the fitted locations of single image point clouds. This can either be found by calculating the mean of all the single image point cloud points or just take the UAV positions for each of these single image point

clouds. Also applying the Block 1 calculated rigid body transform on the recorded UAV position gives us the feature matched single image point cloud's UAV position. These UAV positions have the same characteristics as the feature matched point cloud. Therefore, fitting these UAV positions over GPS recorded UAV positions, will give us a transformation which will be similar to transformation we get from fitting feature matched point cloud over GPS-IMU Pose point cloud. This has been found experimentally also upto 3 decimal places in a small set of recorded readings. A visual overlay of this step is shown in Figure 27.

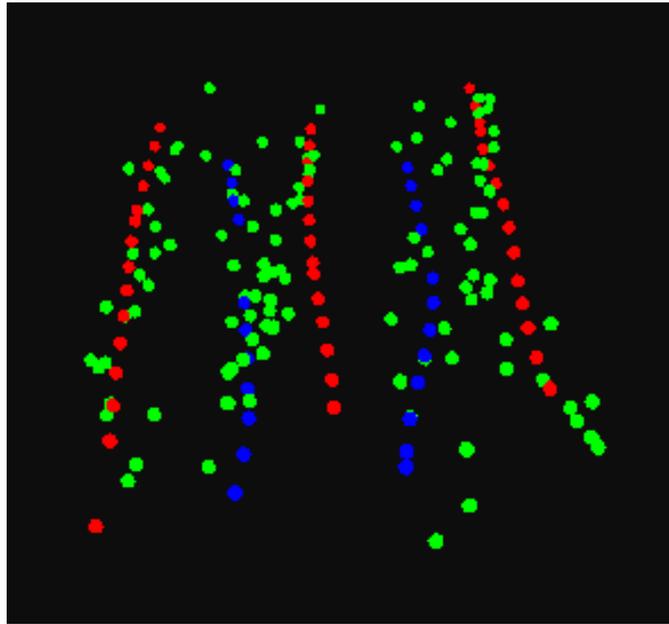


Figure 27: Feature matched UAV positions, shown in green, fitted over GPS recorded UAV positions, shown in red and blue as sequential rows of Lawn Mower UAV path

Table 6 compares the benefits and drawbacks of the chosen correction step, with correction step in Sequential Localization and Mapping (SLAM) algorithm.

Table 6: Comparison of Feature Matching + ICP based correction step with SLAM

Orientation Correction using ICP	State estimation and correction using SLAM
<b>Benefits:</b> <ul style="list-style-type: none"> <li>• Computationally inexpensive</li> <li>• Very low local error</li> </ul>	<b>Benefits:</b> <ul style="list-style-type: none"> <li>• Localization error stays under bounds</li> </ul>

Drawbacks: <ul style="list-style-type: none"> <li>• Localization error propagation over time</li> </ul>	Drawbacks: <ul style="list-style-type: none"> <li>• Computationally expensive</li> <li>• Requires tuning to keep local errors low</li> </ul>
---	--

#### 5.4 Bad Data

An issue found in the image capturing system was production of blurred images after a set of good sharp images. Use of these blurred images in the stereo algorithm, produced bad disparity images. Use of these images used to affect the accuracy of feature matching by a big margin. The effect of these bad images was found after doing feature matching only between 2 sequential images. The result of this is shown in Figure 28. A solution to identify these bad images was found by calculating the variance of all the pixels in a disparity image. It was experimentally found that good disparity images had variance under 3 while bad ones had in range of 10 to 1000. A threshold of 5 was found to be a good threshold to reject bad images.

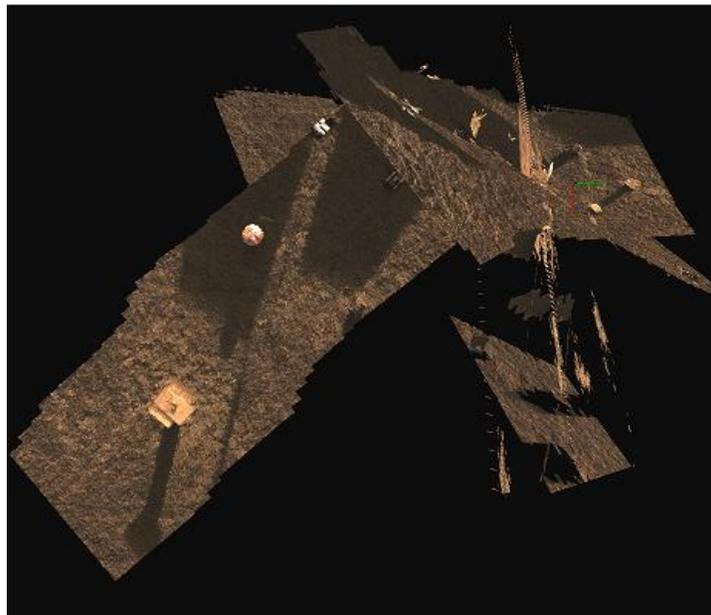


Figure 28: Effect of bad images on sequential feature matched point cloud

## 5.5 Transformation from Image Coordinate to World Coordinates

Figure 29 shows the coordinate transformations from Camera coordinates on UAV to UAV GPS to World Coordinates. Figure 30 shows a method employed to graphically find out angular misalignment in roll and pitch of stereo boom with the UAV. The distance between camera center on the boom and UAV center is 4 cm. UAV center to UAV GPS is 35 cm.

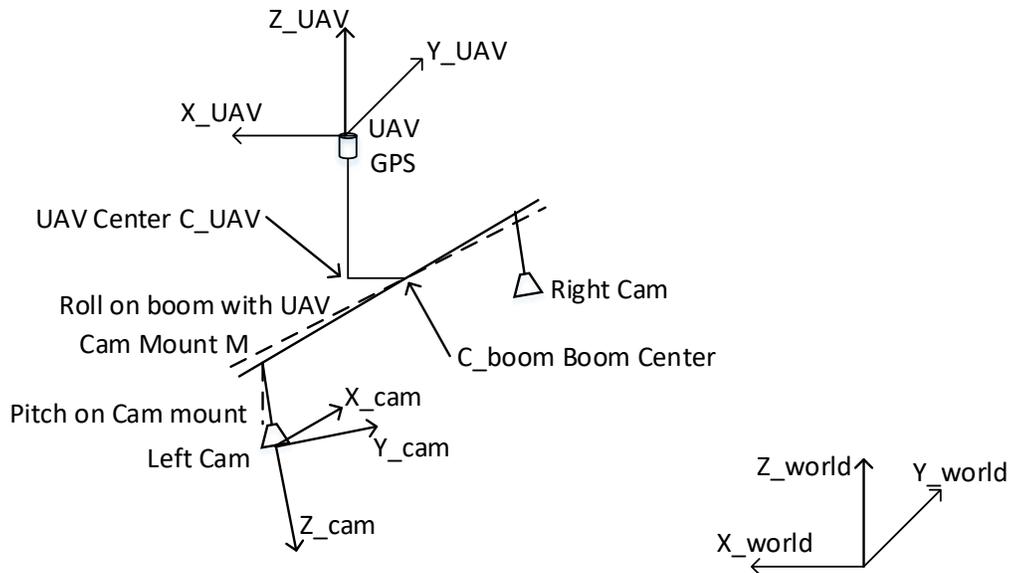


Figure 29: The coordinate transformations from Camera Coordinates to World Coordinates

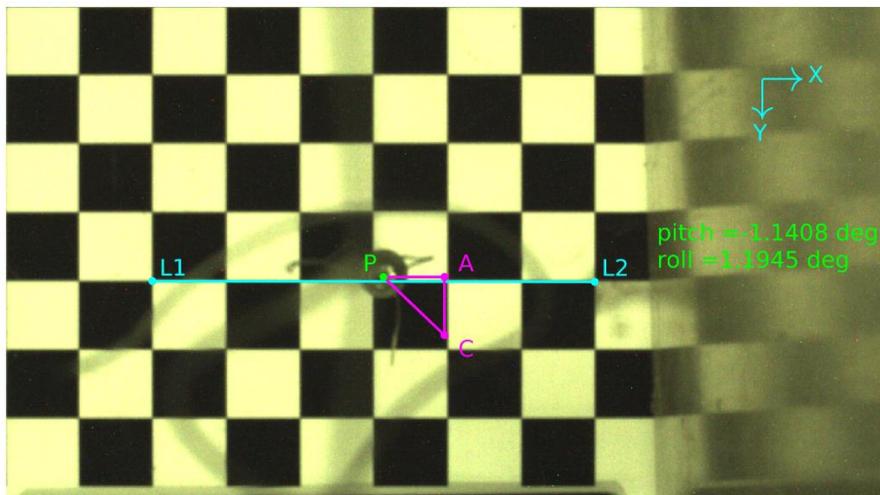


Figure 30: Graphically finding angle of misalignment on stereo boom in roll and pitch

The transformations are done in the following process:

1. Rotation of camera coordinate to account for pitch of stereo boom:  $r_{xi}$
2. Rotation of camera coordinate to account for roll of stereo boom :  $r_{yi}$
3. Inversion of Y-axis of camera coordinate to point it forward and inversion of Z-axis to make all points have negative z-values:  $r_{invert\_i}$
4. Translation from camera coordinate to UAV GPS:  $t_{hi}$
5. Rotation to flip X and Y axis:  $r_{flip\_xy}$
6. Rotation to invert Y axis:  $r_{invert\_y}$
7. Rotation and translation to convert from UAV GPS coordinate system to world coordinate system using the GPS-IMU pose:  $t_{wh}$  and  $r_{wh}$

Using the above, the transformation from camera coordinates to world coordinates is:

$$t_{mat} = t_{wh} * r_{wh} * r_{invert\_y} * r_{flip\_xy} * t_{hi} * r_{invert\_i} * r_{yi} * r_{xi} \quad ( 12 )$$

Naming convention of  $i$  was used incorrectly on the program for camera coordinates, incorrectly calling them image coordinates initially. For consistency purpose, same naming convention is followed here also and  $i$  is used instead of  $c$  as subscript. The function implementing this on the 3D reconstruction program is `generateTmat()`.

## 5.6 Converting Disparity to Depth:

Inputs:

1. Disparity image: each pixel is the disparity in stereo corresponding to that pixel location
2. Q matrix: a 4x4 disparity-to-depth mapping matrix which is an output of OpenCV's stereo calibration result

Output:

XYZ point cloud corresponding to each pixel in the disparity image

Process:

Q is in the form:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix}$$

Where  $c_x$  and  $c_y$  are coordinates of principal point in the left camera.  $c'_x$  is the  $x$  coordinate of the principal point in the right camera. For zero disparity of principal points  $c_x$  and  $c'_x$  will be same. In the built stereo system  $c_x$  and  $c'_x$  are aligned so  $Q(3,3) = 0$ .  $f$  is focal length in pixels.  $T_x$  is the baseline of the stereo cameras.

For location  $(x,y)$  in the disparity image with  $d$  disparity, the 3D point in the camera coordinates will be: Let a vector  $v$  be

$$v = \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}$$

Let  $v$  be  $Q$  multiplied with  $v$ :

$$v = Q * v$$

and normalize by dividing with  $v(3)$ . The 3D projected point  $(X, Y, Z)$  for  $(x,y)$  image location with  $d$  disparity is:

$$X = v(0) = (c_x - x) * T_x / d$$

$$Y = v(1) = (c_y - y) * T_x / d$$

$$Z = v(2) = f * T_x / d$$

This  $(X,Y,Z)$  is in left camera coordinate frame.

## 5.7 Generating low computation cost semi-dense point clouds by jumping over pixels

One of the highest computation step in the process is generation of single image point clouds. Every image has 1.3 megapixels. This step can smartly be reduced in time by reducing the number of pixels being used. Instead of reading every pixel, every 5<sup>th</sup> or 10<sup>th</sup> pixel in row and column can be read. This will generate a semi-dense point cloud. For many applications a dense cloud is not required, therefore, this is a useful step for fast 3D reconstruction. In the program this is controlled

by the variable `jump_pixel`. A value of 1 means all pixels are read. A comparison of different `jump_pixel` values is shown in Figure 31.

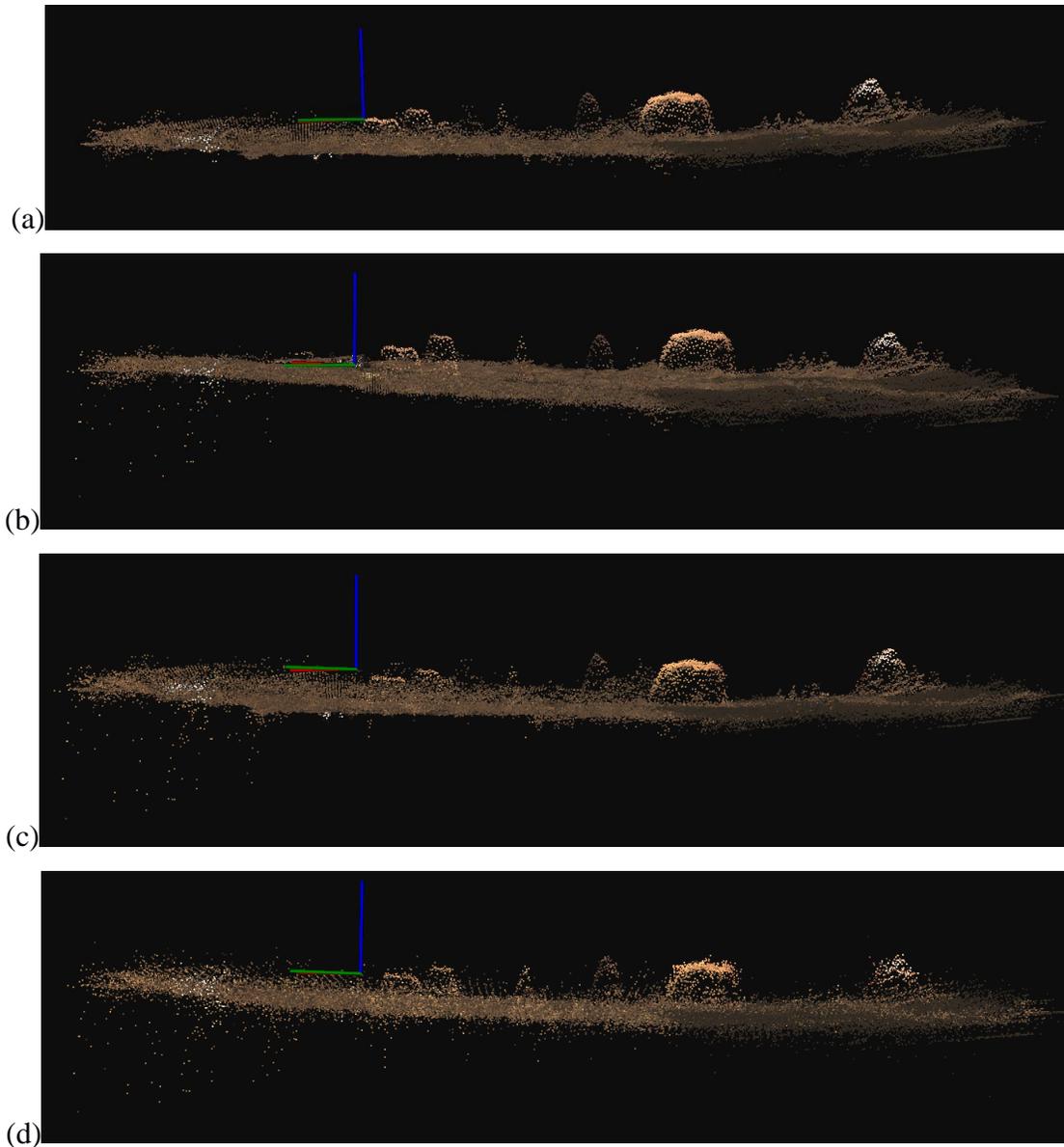


Figure 31: Comparison of Point Clouds generated using jump pixels of (a) 1 pixel (b) 2 pixel (c) 5 pixel and (d) 15 pixel

## 5.8 Ground Segmentation using Sample Consensus Plane Fitting

An aim of the project is to produce a traversable area for a UGV on the ground. This is done by ground segmentation. The process employed to achieve this is described in this section. First the point cloud is divided into smaller areas of  $4 \times 4 \text{ m}^2$  as shown in Figure 32.

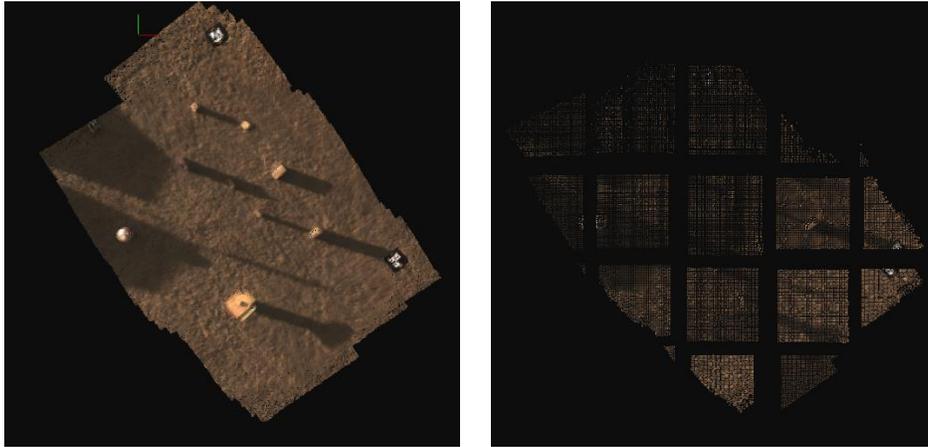


Figure 32: Dividing generated 3D reconstruction into smaller blocks

Over these smaller blocks a RANSAC sample consensus plane fitting is run for all points at a distance of a threshold value, e.g., 10 cm. The best fit planes from all the blocks are then combined and a ground plane generated. Over this ground plane a Convex Hull fitting is done to generate its boundary points. Both these results are shown in Figure 33.

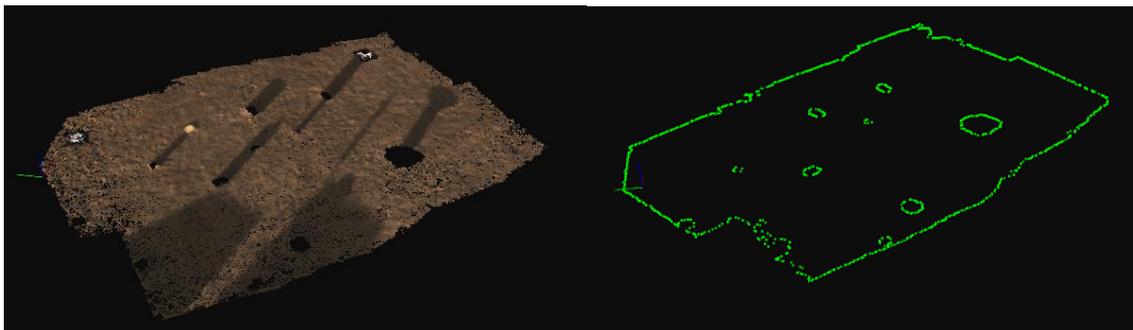


Figure 33: Segmented ground with Convex hull fitted boundary locations

By dividing the point cloud into smaller regions, a smaller value of distance threshold is used for RANSAC plane fitting. The curvature of ground remains conserved in this process. A potential failure of this process can be in sudden and long change in heights of terrain, e.g., roof of a house or a big ditch. Potential ways to tackle these situations is to match edge heights of fitted planes before combining them back to get segmented ground. Planes significantly away from neighbors can be marked as obstacles themselves.

## Chapter 6: Results

### 6.1 Experiment Region

The obstacle course to 3D reconstruct was setup at Kentland Farms. The region of interest was kept as 25 x 40 m<sup>2</sup> region. 9 obstacles were placed on the course. Top view of this course is shown in Figure 34. This mosaic was created by using the captured images on Agisoft PhotoScan 3D reconstruction software. Using surveying equipment, GPS locations of each of the obstacles was recorded. The perspective view of this 3D reconstruction is shown in Figure 35.

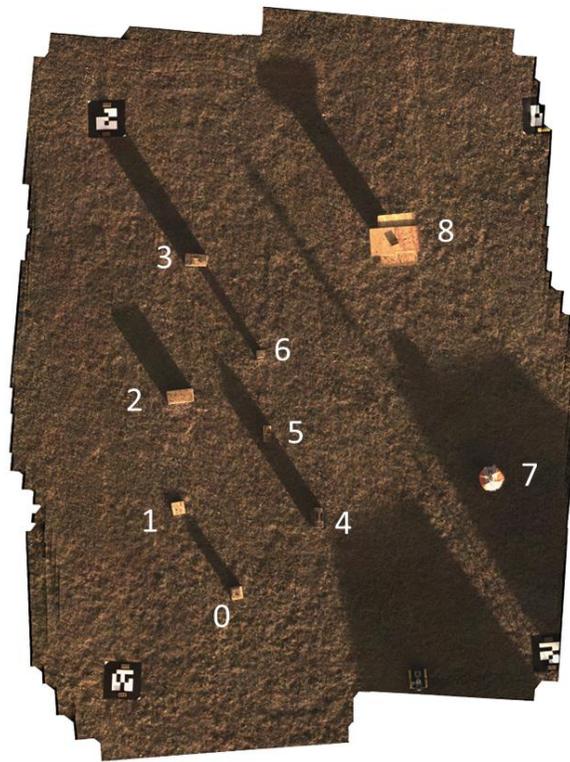


Figure 34: Obstacle course setup with 9 obstacles. Image mosaic created using Agisoft PhotoScan software

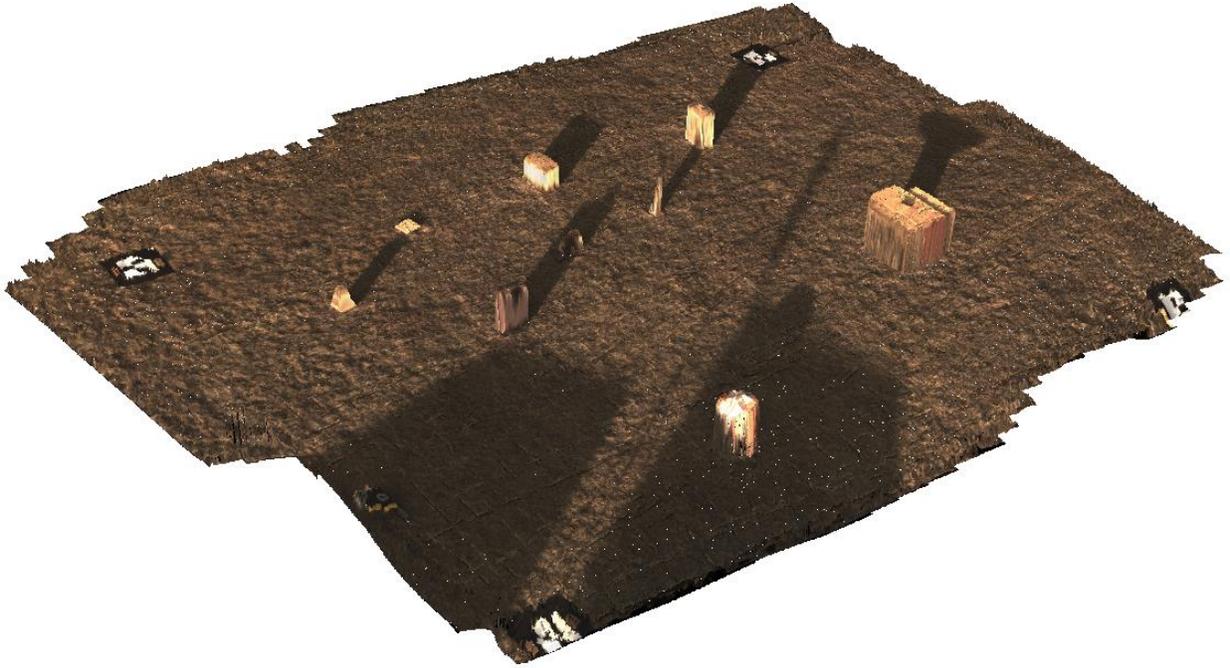


Figure 35: 3D reconstruction using Agisoft PhotoScan software.

The surveying equipment was separate from the GPS on the UAV and it was found that there is an offset in origins of coordinate values reported by UAV GPS and the surveying equipment GPS on the ROS system. This was found in data post-processing and a reason for this has not been deciphered at the writing of this thesis. The offset is calculated as an average mismatch in obstacle localizations using 3D reconstruction and the surveyed locations. As of now the reason appears to be hardware or ROS issue, therefore, this offset is removed while reporting the results. If it is a calculation error then the error would be in Section 5.5 where transformations have been calculated to convert image coordinates to world coordinates. This offset is  $(-7.16 \text{ m}, -5.87 \text{ m})$ . Besides this another offset in the z direction of 1 m has been applied for the same reason.

## 6.2 3D Reconstruction Results

The obstacle ground truth and 3D reconstruction dimensions and locations are mentioned in Appendix A. Figure 36 shows the perspective view of dense 3D reconstruction result. This figure can be compared to Figure 35 for a visual comparison of 3D reconstruction using the presented approach. Figure 37 compares reconstructed z-heights of obstacles with recorded z-heights. This z-height is height above the base of the base station GPS antenna. Figure 38 shows localization accuracy by overlaying obstacle positions in reconstructed results over recorded positions.

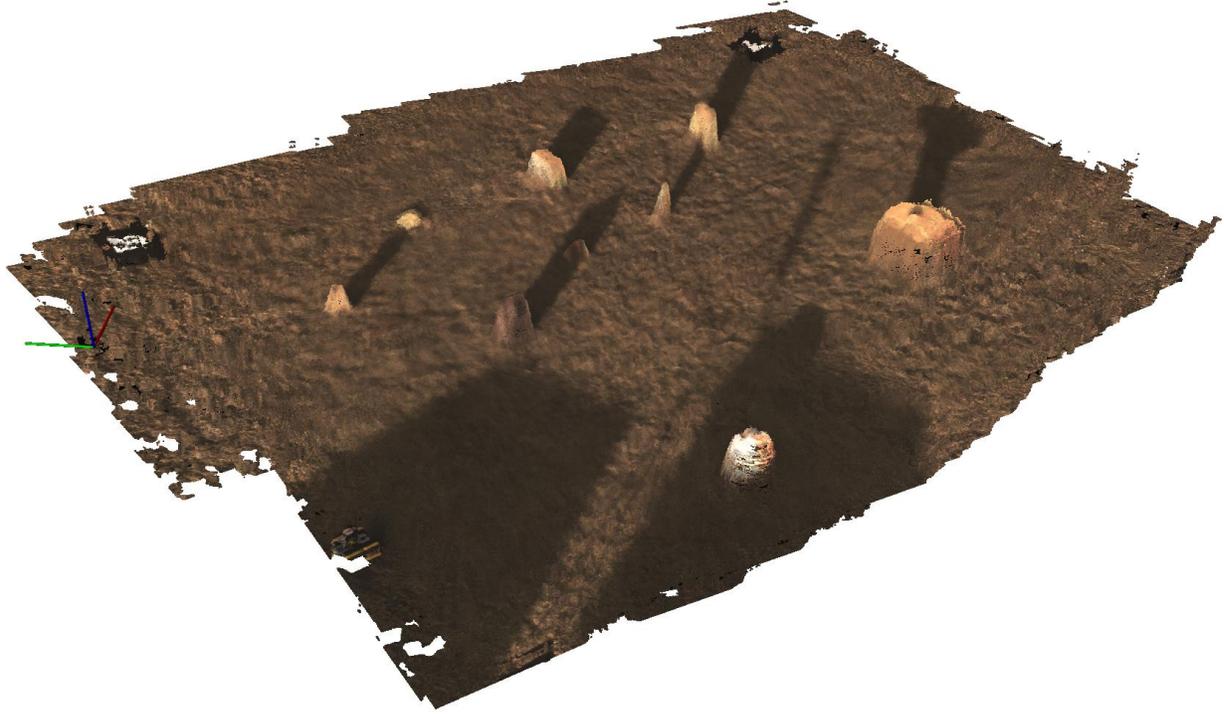


Figure 36: Dense 3D reconstruction

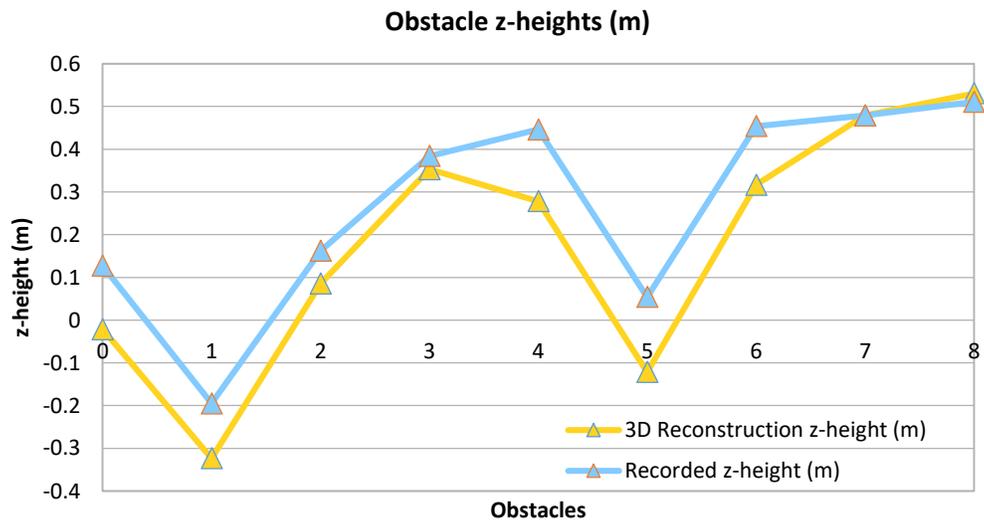


Figure 37: Comparison of obstacle ground truth altitudes with 3D reconstruction results

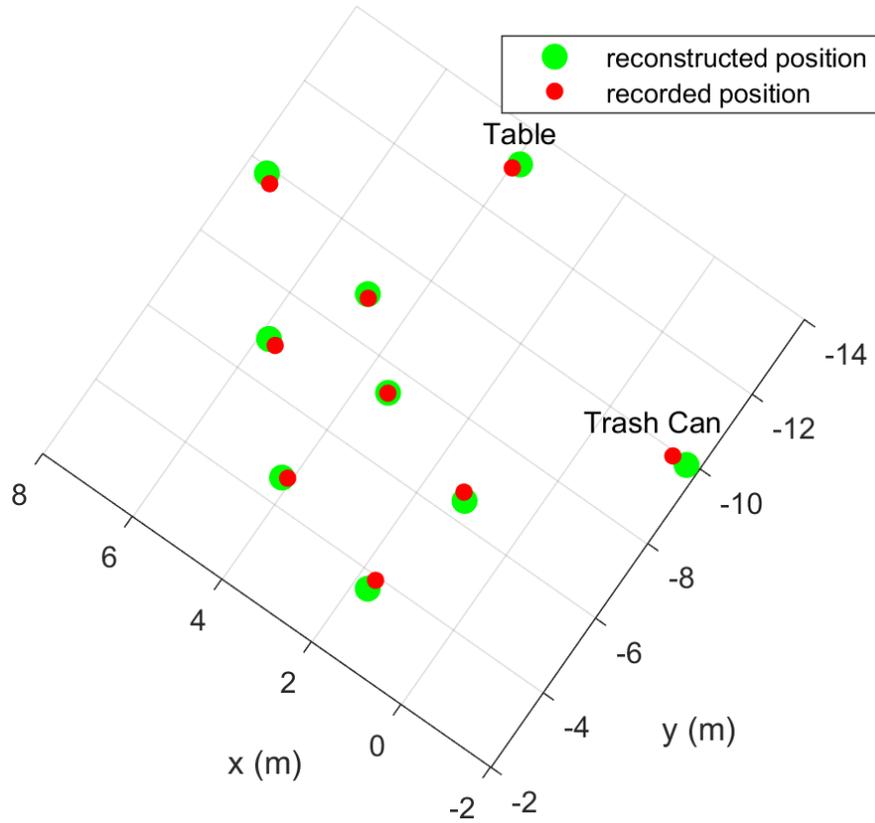


Figure 38: Localization accuracy visualization

Table 7 lists the error in position and z-height of obstacles. Table 8 lists the average and standard deviation error in localization, z-height and dimensions of obstacles.

Table 7: Error in position and altitude of obstacles in 3D reconstruction

Obstacle	Error in Position (m)	Error in z-Height (m)
0	0.25	0.15
1	0.11	0.13
2	0.17	0.08
3	0.21	0.03
4	0.18	0.17
5	0.01	0.18

6	0.09	0.14
7	0.30	0.00
8	0.18	-0.02

Table 8: Average and standard deviation in localization, altitude and dimension error of obstacles in 3D reconstruction

Obstacle Localization Error		Obstacle z-height Error		Obstacle Size Error	
Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
17 cm	9 cm	9 cm	7 cm	3 cm	7 cm

### 6.3 Stereo Vision System Localization and Reconstruction Accuracy

As inferred from Table 8, the stereo vision system and 3D reconstruction process generates results which have localization error and standard deviation of 17 cm and 9 cm, respectively. The process is able to estimate obstacles sizes with error and standard deviation of 3 cm and 7 cm, respectively.

### 6.4 3D Reconstruction Speeds

To judge the speeds in steps of program, the program is divided into 5 blocks:

- Finding features
- Matching features
- ICP Correction
- Point Cloud Creating
- Downsampling

The running times of the program is compared with change in two parameters:

- jump\_pixels
- voxel\_size

The comparison of running speeds of the program in the five blocks with jump\_pixels is shown in Figure 39.

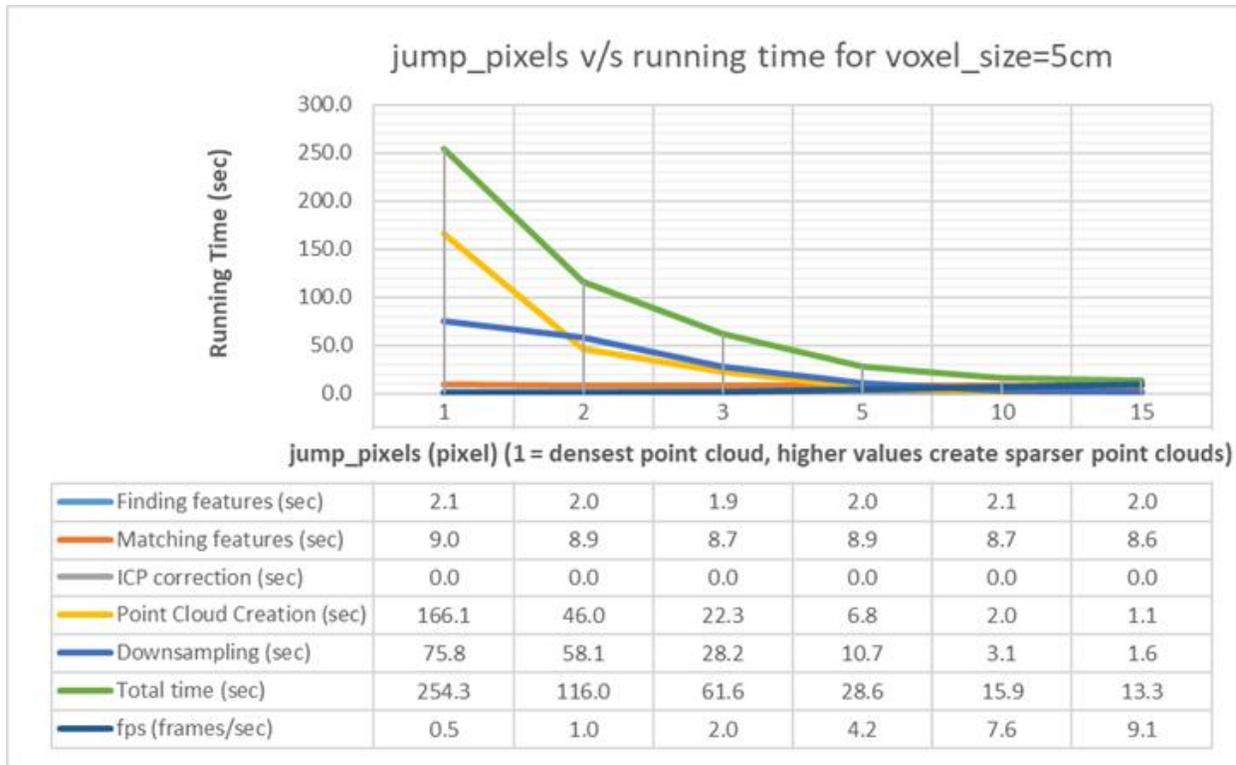


Figure 39: Comparison of program running speeds with jump\_pixels

Analyzing the results we see that most time is consumed at Downsampling and Point Cloud Creation steps. The jump\_pixels value controls the density of the point cloud being generated. We see that it produces an exponential decrease in running time. With jump\_pixel of 1 the point cloud creation and downsampling times were 166.1 sec and 75.8 sec, respectively, which on using jump\_pixels value 15 get reduced to 1.1 sec and 1.6 sec, respectively. We see that finding feature time and feature matching time understandable remains invariant to jump\_pixels because they work on keypoints rather than the whole image. After jump\_pixels value 5, the feature matching on GPU becomes the bottleneck in the program. It takes a total of 8.6 sec out of 13.3 sec total time in jump\_pixels 15, which is 65% of the program time. This says that using the current functions of OpenCV Feature Matching on GPU, we are approaching to the limits of how good the program can be. To improve the program further, a new matching function step needs to be implemented to reduce this 8.6 sec time. We see the speed of the program with jump\_pixels 15 is 9.1 fps for voxel\_size 5 cm.

Figure 40 compares the running times of program with different values of voxel\_size. We see that understandably only downsampling time has some change whereas all other steps take the same time because voxel\_size acts only at that step. An interesting inference to note is that it has only a minor change on the downsampling size, from 12.9 sec to 10.4 sec for voxel\_size of 1 cm to 10 cm, respectively.

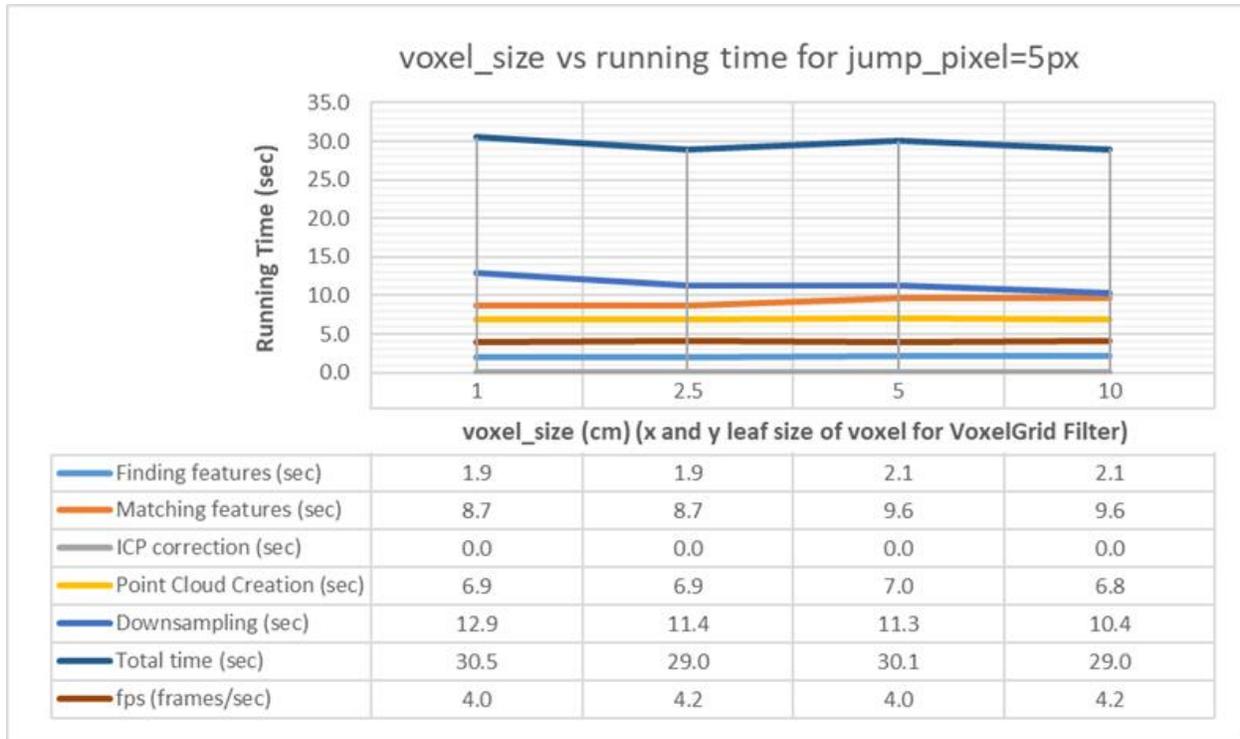


Figure 40: Comparison of program running speeds with voxel\_size parameter

### 6.5 Effect of Different Approaches on 3D Reconstruction

An important understanding of how 3D reconstruction results are affected by different approaches is described in this section. This was the step by step process we implemented to understand what was happening improve the 3D reconstruction of long range stereo data. Figure 41 is 3D reconstruction generated using raw stereo output and raw GPS-IMU Pose. Since outliers haven't been removed or the discontinuous stepped output of raw stereo hasn't been adjusted for, the point cloud looks highly disfigured and noise is very high. Also, understandably, as with any sensor, there is sensor noise in GPS localization and Pixhawk UAV state estimation. This noise on being transformed to 23 m away, creates a huge local localization error in the results. Downsampling

this point cloud using the Voxel Grid filter, blurs out the obstacles and gives a very inaccurate 3D reconstruction result.

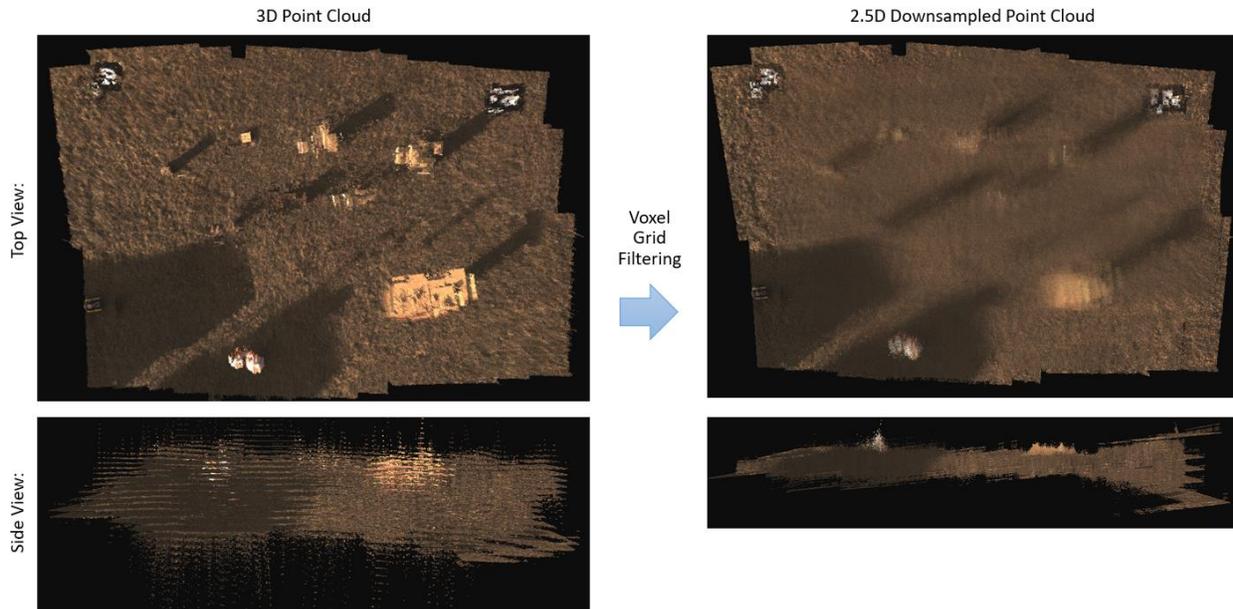


Figure 41: 3D reconstruction using Raw GPS-IMU Pose and Raw Stereo Output

Figure 42 shows the point cloud created using the presented clustering process. We see that because of this the noise in the generated point cloud has gone. Another thing we notice is a tilt in images. This is most probably because the UAV is also tilted a bit while moving ahead. Also the stereo boom has around a 1.5 deg misalignment with the UAV. Still the result is not good but using this clustering process we were able to understand what was happening in the 3D reconstruction and that we needed a lot of filtering steps to clean the noise from raw stereo output.

Figure 43 shows 3D reconstruction results using the clustering process along with feature matching. We see the results are really good visually. The obstacles have aligned well and localization seems to be good. One thing noticed here was that clustering process sometimes misses out on fitting small objects inside its cluster. This blurs out the obstacle with the ground. Therefore, even though the localization is good, the obstacle heights are way off the ground truth. Another benefit of this clustering process was the help it provided to improve the feature matching process by visually helping in development.

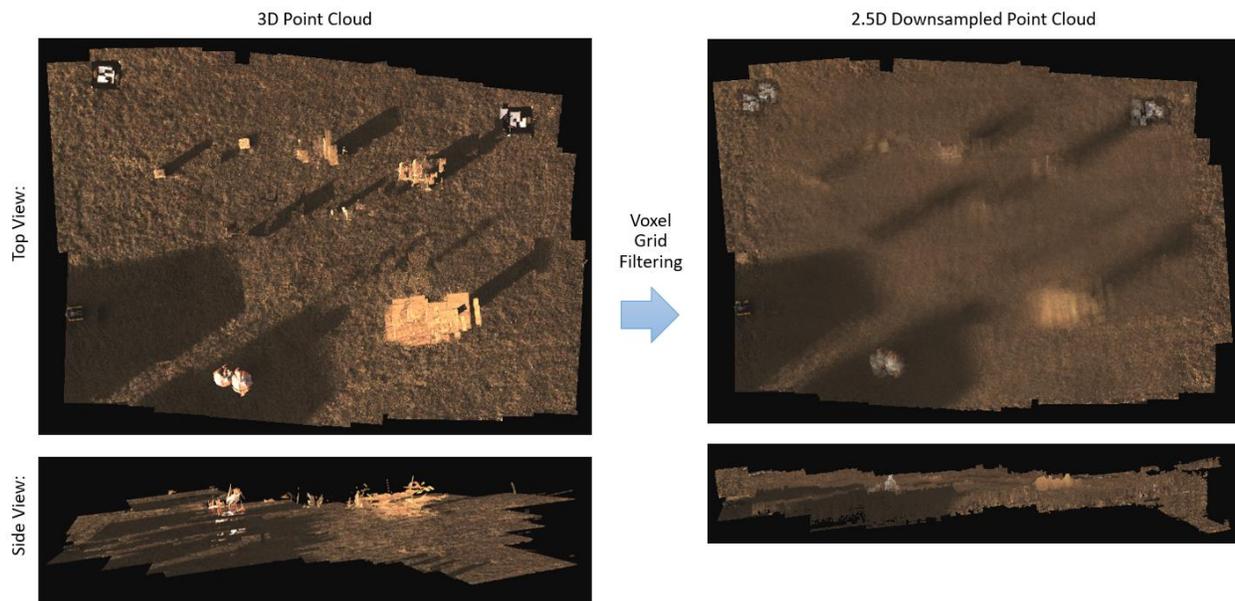


Figure 42: 3D reconstruction using Clustering Process to improve disparity image resolution and noise removal and Raw GPS-IMU Pose

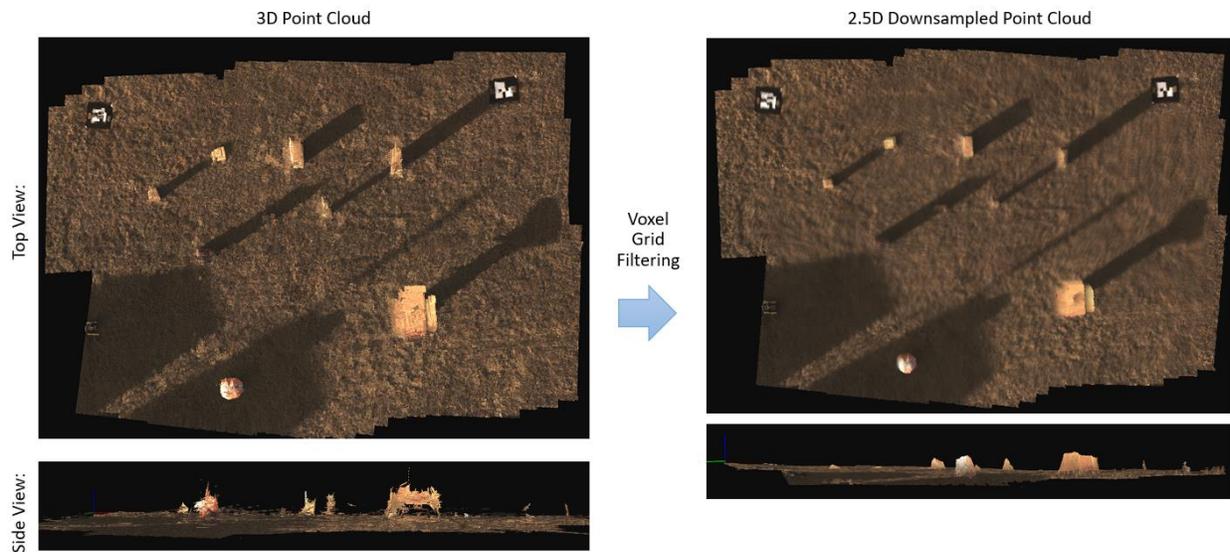


Figure 43: 3D reconstruction using Clustering Process to improve disparity image resolution and noise removal and Feature Matching Process

Figure 44 shows the 3D reconstruction process using filtering steps to remove noise and feature matching to align images. As seen, the localization is good as well as the z-heights of obstacles near to the ground truth.

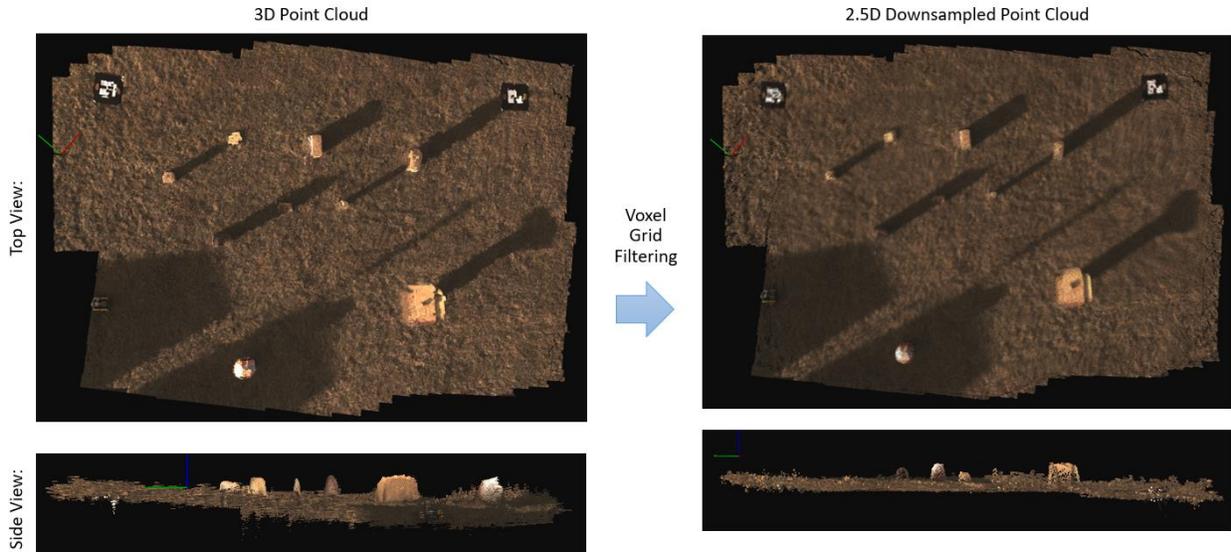
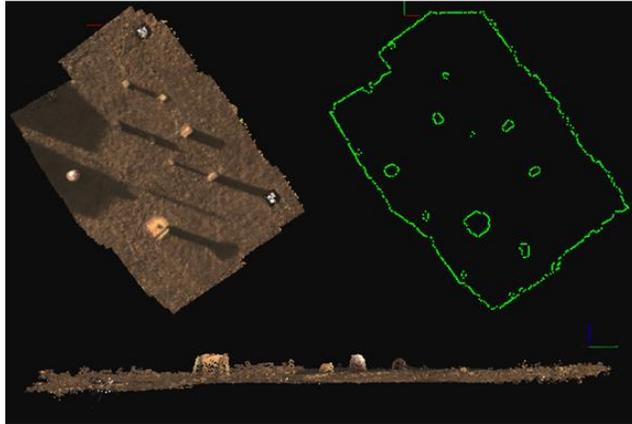


Figure 44: 3D reconstruction using Filtering methods for noise removal and Feature Matching Process

## 6.6 Ground Segmentation Results

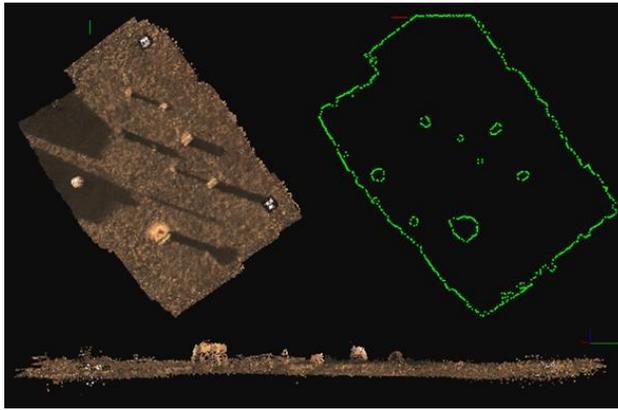
To test the ground segmentation with different sparsity levels in point clouds, we compare the output of segmentation at different values of `jump_pixels`. Figure 45 and Figure 46 show results of ground segmentation for different values of `jump_pixels` with `voxel_size` 5 cm and 10 cm, respectively. We note that in 5 cm `voxel_size`, even though the point cloud is becoming sparser, the results of the ground segmentation do not decrease. It still is able to capture and remove most obstacles. With 10 cm `voxel_size` we see that with increase in `jump_pixels`, the noise in segmentation increases. An important inference to note here is that noise is mostly higher at the edges of the region of interest. At center locations, there are more images and, therefore, more data, thereby making the ground segmentation much better compared to near edge areas.

jump\_pixels 5 @ 2.3 fps



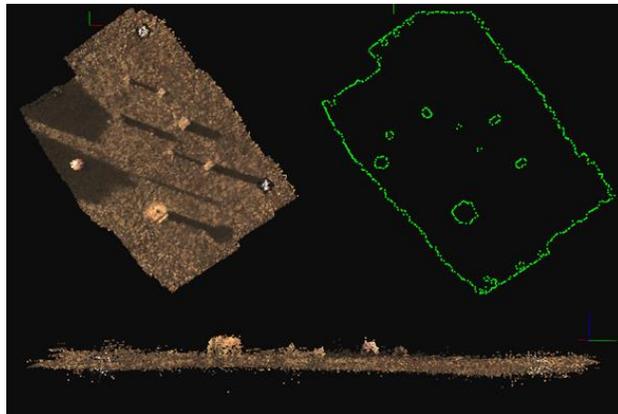
(a)

jump\_pixels 10 @ 7.2 fps



(b)

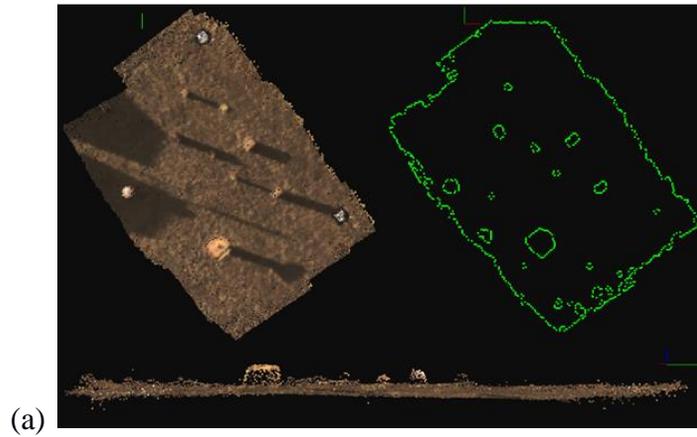
jump\_pixels 15 @ 10.2 fps



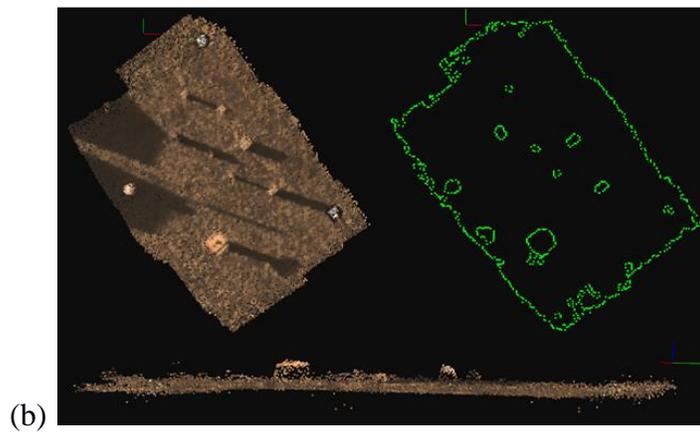
(c)

Figure 45: Comparison of ground segmentation results with different values of jump\_pixels for voxel\_size 5 cm

jump\_pixels 10 @ 7.4 fps



jump\_pixels 15 @ 10.3 fps



jump\_pixels 25 @ 13.1 fps

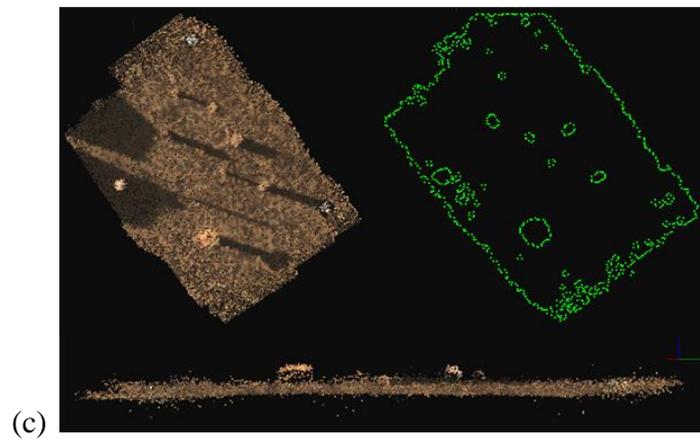


Figure 46: Comparison of ground segmentation results with different values of jump\_pixels for voxel\_size 10 cm

## Chapter 7: Conclusion and Future Work

### 7.1 Conclusion

The thesis presents a fast way to do accurate and online 3D reconstruction and ground segmentation using drone based stereo vision with a novel correction step. The presented process gives high localization as well as object size/dimensional accuracy and is able to run at 8-15 fps. The built stereo vision system is low cost. The long baseline stereo gives better resolution at higher altitudes compared to market solutions which generally have smaller baseline.

A novel way of using 2D texture in a clustering process to produce high resolution of disparity images is presented. The depth values remain conserved with this process but the stepped discontinuity in stereo results gets transformed to clean continuous results.

### 7.2 Future Work

This section presents some of the potential improvements which can be done in the process to further improve results and increase applications:

- Integrating the program with USL's ROS based multi-robot control system
- Extending the work to make a generalized 3D reconstruction solution rather than a UAV based ground 3D reconstruction solution
- Use of Kalman Filter for sensor fusion of GPS/IMU localization with Feature Matching Localization
- Ground 3D reconstruction strategies other than Lawn Mower Pattern Paths
- Bundle Adjustment to improve Feature Matching Accuracy
- Extensibility to large areas easily achievable by leaving well fitted point cloud out of further fitting computations
- Testing system with more data to figure out unfound types of bad data

## References

- [1] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, “LayoutNet: Reconstructing the 3D Room Layout from a Single RGB Image,” 2018.
- [2] M. Miknis, R. Davies, P. Plassmann, and A. Ware, “Efficient Point Cloud Pre-processing using The Point Cloud Library,” *Int. J. Image Process.*, no. 10, pp. 63–72, 2016.
- [3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2564–2571, 2011.
- [4] E. Karami, S. Prasad, and M. Shehata, “Image Matching Using SIFT , SURF , BRIEF and ORB : Performance Comparison for Distorted Images Image Matching Using SIFT , SURF , BRIEF and ORB : Performance Comparison for Distorted Images,” *Newfoundl. Electr. Comput. Eng. Conf.*, no. November, p. 4, 2015.
- [5] “cv::cuda::DescriptorMatcher Class Reference.” [Online]. Available: [https://docs.opencv.org/3.1.0/dd/dc5/classcv\\_1\\_1cuda\\_1\\_1DescriptorMatcher.html](https://docs.opencv.org/3.1.0/dd/dc5/classcv_1_1cuda_1_1DescriptorMatcher.html).
- [6] J. Heikkila and O. Silven, “A four-step camera calibration procedure with implicit image correction,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1106–1112.
- [7] J. Gassaway, K. Kochersberger, A. L. Abbott, M. Roan, and J. Gassaway, “Local Bundling of Disparity Maps for Improved Dense 3D Visual Reconstruction,” 2011.
- [8] H. Hirschm, “Stereo Processing by Semi-Global Matching and Mutual Information,” pp. 1–14, 2007.
- [9] D. Hernandez-Juarez, A. Chacon, A. Espinosa, D. Vazquez, J. C. Moure, and A. M. Lopez, “Embedded real-time stereo estimation via Semi-Global Matching on the GPU,” *Procedia Comput. Sci.*, vol. 80, pp. 143–153, 2016.
- [10] R. Kalarot and J. Morris, “Analysis of real-time stereo vision algorithms on GPU,” ... *Image Vis. ...*, no. June, pp. 1–6, 2011.
- [11] B. Triggs, P. F. P. McLauchlan, R. I. Hartley, and A. W. A. Fitzgibbon, “Bundle Adjustment—A Modern Synthesis,” *Cs.Jhu.Edu*, vol. 34099, pp. 298–372, 2000.
- [12] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping (SLAM): part I The Essential Algorithms,” *Robot. Autom. Mag.*, vol. 2, pp. 99–110, 2006.
- [13] H. Shah-hosseini, “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods,”

*IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 10, pp. 1388–1393, 2002.

Appendix A: Ground truth and 3D Reconstruction results comparison of obstacle localization and size

<b>0_Three box stack</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.41	0.41	0.00
length	0.22	0.35	0.13
width	0.21	0.26	0.05
x	1.68	1.71	0.04
y	-3.91	-3.66	0.24
z	0.13	-0.02	-0.15
<b>1_small wooden single box</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.10	0.04	-0.06
length	0.31	0.42	0.11
width	0.31	0.39	0.08
x	4.07	4.16	0.09
y	-4.64	-4.58	0.06
z	-0.20	-0.32	-0.13
<b>2_cardboard box medium size</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.46	0.39	-0.07
length	0.62	0.70	0.08
width	0.32	0.37	0.05
x	5.65	5.81	0.16
y	-6.87	-6.91	-0.04
z	0.16	0.09	-0.08
<b>3_cardboard box big size</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.78	0.76	-0.02
length	0.51	0.62	0.11
width	0.28	0.33	0.05
x	7.43	7.58	0.14
y	-9.71	-9.86	-0.15
z	0.38	0.35	-0.03

<b>4_suitcase</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.73	0.81	0.08
length	0.48	0.49	0.01
width	0.30	0.30	0.00
x	1.28	1.18	-0.10
y	-6.61	-6.46	0.14
z	0.45	0.28	-0.17
<b>5_Two cardboard vertical stack</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.37	0.34	-0.03
length	0.40	0.43	0.03
width	0.18	0.20	0.02
x	3.46	3.46	0.00
y	-7.43	-7.44	0.00
z	0.05	-0.12	-0.18
<b>6_Vertical thin stack</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.80	0.80	0.00
length	0.18	0.24	0.06
width	0.15	0.21	0.06
x	4.75	4.80	0.05
y	-8.89	-8.96	-0.07
z	0.45	0.32	-0.14
<b>7_Trash can</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
height	0.80	0.73	-0.07
diameter	0.60	0.73	0.13
x	-1.47	-1.77	-0.30
y	-9.89	-9.90	-0.01
z	0.48	0.48	0.00
<b>8_Table</b>	<b>Ground truth (m)</b>	<b>Reconstructed (m)</b>	<b>Error (m)</b>
h1	0.93	0.82	-0.11
h2	0.84	0.84	0.00
length	1.24	1.38	0.14
width	1.29	1.33	0.04
x	3.96	3.88	-0.08

y	-13.05	-13.21	-0.16
z	0.51	0.53	0.02