

Semi-Dense Stereo Reconstruction from Aerial Imagery for Improved Obstacle Detection

James J. Donnelly

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
Mechanical Engineering

Kevin K. Kochersberger, Chair

Alfred L. Wicks

Pratap Tokekar

September 16, 2019

Blacksburg, Virginia

Keywords: UAV, Stereo Vision, 3D Reconstruction, Mapping

Copyright 2019, James J. Donnelly

Semi-Dense Stereo Reconstruction from Aerial Imagery for Improved Obstacle Detection

James J. Donnelly

(ABSTRACT)

Visual perception has been a significant subject matter of robotics research for decades but has accelerated in recent years as both technology and community are more prepared to take on new challenges with autonomous systems. In this thesis, a framework for 3D reconstruction using a stereo camera for the purpose of obstacle detection and mapping is presented. In this application, a UAV works collaboratively with a UGV to provide high level information of the environment by using a downward facing stereo camera. The approach uses frame to frame SURF feature matching to detect candidate points within the camera image. These feature points are projected into a sparse cloud of 3D points using stereophotogrammetry for ICP registration to estimate the rigid transformation between frames. The RTK-GPS constrained pose estimate from the UAV is fused with the feature matched estimate to align the reconstruction and eliminate drift. The reconstruction was tested on both simulated and real data. The results indicate that this approach improves frame to frame registration and produces a well aligned reconstruction for a single pass compared to using the raw UAV position estimate alone. However, multi-pass registration errors occur on the order of about 0.6 meters between parallel passes, and approximately 2 degrees of local rotation error when compared to a reconstruction produced with Agisoft Metashape. However, the proposed system performed at an average frame rate of about 1.3 Hz compared to Agisoft at 0.03 Hz. Overall, the system improved obstacle registration and can perform online within existing ROS frameworks.

Semi-Dense Stereo Reconstruction from Aerial Imagery for Improved Obstacle Detection

James J. Donnelly

(GENERAL AUDIENCE ABSTRACT)

Visual perception has been a significant subject matter of robotics research for decades but has accelerated in recent years as both technology and community are more prepared to take on new challenges with autonomous systems. In this thesis, a framework for 3D reconstruction using cameras for the purpose of obstacle detection and mapping is presented. In this application, a UAV works collaboratively with a UGV to provide high level information of the environment by using a downward facing stereo camera. The approach uses features extracted from camera images to detect candidate points to be aligned. These feature points are projected into a sparse cloud of 3D points using stereo triangulation techniques. The 3D points are aligned using an iterative solver to estimate the translation and rotation between frames. The RTK (Real Time Kinematic) GPS constrained position and orientation estimate from the UAV is combined with the feature matched estimate to align the reconstruction and eliminate accumulated errors. The reconstruction was tested on both simulated and real data. The results indicate that this approach improves frame to frame registration and produces a well aligned reconstruction for a single pass compared to using the raw UAV position estimate alone. However, multi-pass registration errors occur on the order of about 0.6 meters between parallel passes that overlap, and approximately 2 degrees of local rotation error when compared to a reconstruction produced with the commercial product, Agisoft. However, the proposed system performed at an average frame rate of about 1.3 Hz compared to Agisoft at 0.03 Hz. Overall, the system improved obstacle registration and can perform online within existing Robot Operating System frameworks.

Dedication

To my wife, who supported me throughout the long journey of completing both my undergraduate and graduate degree. Her unending patience, help, and dedication to our goals kept me centered, focused, and inspired through the years.

Acknowledgments

I would like to extend my tremendous gratitude to all the people who have both helped me get started in this project, as well as walked with me through all the trials along the way. I want to thank Dr. Kevin Kochersberger for giving me this opportunity, and for his support and guidance. I would also like to thank Dr. Alfred Wicks, Dr. Pratap Tokekar, and Dr. John Bird for providing exceptional educational background which I was able to apply throughout this project, as well as for discussing challenges in my work along the way. I would like to thank Prashant Kumar for providing the background I needed in 3D vision and reconstruction to do begin this project. John Peterson, for both developing a robotics system I could learn from and build off of in the lab, as well as being an ever patient resource to discuss both existing algorithms and potential improvements. Anthony Wagner, for being a reliable friend and pilot throughout, and for his improvements to the system overall, especially his work in path planning and exploration. Haseeb Chaudhry, for being super helpful with graduate school planning and for encouraging me along the way. Of course, Drew Morgan of the Unmanned Systems Lab for his positive demeanor, bright personality, and willingness to help in any way he can. Finally, a special thanks to all of my family and friends who have offered nothing but support at every turn.

Contents

List of Figures	x
List of Tables	xiii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Approach	1
2 Review of Literature	3
2.1 Applications	3
2.2 Feature Matching	6
2.2.1 ORB	7
2.2.2 SURF	7
2.3 SLAM	8
2.3.1 ORB-SLAM	9
2.3.2 LSD-SLAM	9
2.4 Implementations	10
3 Background	13

3.1	Project Motivation	13
3.1.1	Obstacle Detection	14
3.2	Computer Vision	15
3.2.1	Camera Model	15
3.2.2	Stereo Vision	17
3.3	Coordinate Frames	20
3.4	Mechanical Design	23
3.4.1	Prototype	23
3.4.2	Design	25
3.5	Environment	27
3.5.1	V-REP	27
3.5.2	MATLAB	28
3.5.3	Test Conditions	30
4	Methodology	33
4.1	Integration	33
4.2	Reconstruction Algorithm Overview	35
4.3	Software	36
4.3.1	Existing Code	37
4.3.2	ROS Integration	37

4.4	Feature Matching	39
4.5	Outlier Filtering	40
4.6	ICP	42
4.6.1	Initialization	43
4.6.2	ICP Biasing	44
4.7	Kalman Filtering	45
4.8	Scene Reconstruction	47
5	Results	48
5.1	Simulation	48
5.1.1	Evolution of State Estimation	48
5.1.2	Reconstruction of Simulated Environment	50
5.2	Real World	54
5.2.1	Scenes Compared	54
6	Discussion	57
6.1	Errors	57
6.1.1	Tracking	57
6.1.2	Position	60
6.2	Agisoft	61
6.2.1	Compared to proposed system	62

6.3 Future Work	64
7 Summary and Conclusions	67
Bibliography	69
Appendices	73
Appendix A Additional Reconstruction Figures	74
Appendix B Mechanical Drawings	76

List of Figures

2.1	Applications of 3D reconstruction	4
2.2	Comparison of laser, photogrammetry, and infrared sensing on two different scenes	5
2.3	The Gaussian derivative is discretized into a 9x9 grid and approximated	8
2.4	Point cloud data (left), is used to generate an octree map (right) showing occupied voxels	10
2.5	Dense 3D reconstruction from aerial stereo imagery	11
3.1	Pinhole Camera Model	15
3.2	Epipolar geometry reduces correspondence complexity to 1D search along epipolar lines	18
3.3	ROS camera calibration interface	19
3.4	Disparity after camera calibration shows a car passing through the scene	19
3.5	Camera coordinate frame convention is x positive along image right	20
3.6	Coordinate frames for the camera and the UAV base link	21
3.7	Prototype mounted to UAS	24
3.8	prototype used 3D printed ABS for camera mounts on a round carbon fiber tube	24
3.9	New stereo camera boom is easier to align and maintains calibration longer	25

3.10	Camera Mounting	26
3.11	Planer alignment achieved with setscrews	26
3.12	V-REP Environment	28
3.13	Close-up of simulated stereo boom and V-REP scene	28
3.14	Lawn-Mower pattern flight plan to collect real data for algorithm testing	31
4.1	Part of the processing performed and development off-board	34
4.2	3D Reconstruction algorithm illustrated as six primary block processes	35
4.3	Previous frame features are depicted by + markers (cyan overlay)	40
4.4	Side by side view of feature matching between frames	40
4.5	Disparity noise	41
4.6	While both images have the same number of strongest features shown (500), (a) was produced using simply the strongest match algorithm	42
4.7	Bias points add weight (favor) the transnational component of the ICP algo- rithm	45
5.1	Estimated position using only feature matching	49
5.2	Estimate with ICP Biasing	50
5.3	Estimate with Kalman Filtering	51
5.4	Reconstruction of simulated environment	52
5.5	Reconstruction accuracy of simulated scene	53
5.6	Ground plane extraction from sim-reconstruction	53

5.7	Agisoft 2D Orthomosaic of images taken during real data collection with labels identifying key objects in the scene	54
5.8	Top view reconstruction	55
5.9	Front view reconstruction	55
5.10	On the existing system using the raw position estimate of the FCU, object registration is poor	56
6.1	When turning to begin the next pass, tracking is lost and results in a shift relative to the previous pass	58
6.2	Data from each pass is used to reconstruct the scene segments	59
6.3	The complete scene reconstruction demonstrates tracking errors which lead to reduced registration quality	59
6.4	Position error between ground truth (UAV position) and the estimated position	60
6.5	Agisoft tracking issues in grassy region between buildings	62
6.6	Reconstruction produced using Agisoft	63
6.7	Reconstruction produced using Agisoft compared to the results produced from the proposed algorithm	64

List of Tables

3.1	Test hardware specifications	32
6.1	Magnitude of mean position error between ground truth and position estimate	61

List of Abbreviations

COTS Commercial Off The Shelf

FCU Flight Control Unit

FOV Field of View

GPS Global Positioning System

ICP Iterative Closest Point

ORB Oriented FAST and rotated BRIEF

ROS Robot Operating System

RTK Real Time Kinematic

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SURF Speeded Up Robust Features

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

Chapter 1

Introduction

1.1 Background and Motivation

With the advancement of Unmanned Aerial System (UAS) technology, it is becoming increasingly feasible to integrate UASs in a variety of tasks. The specific application presented in this thesis, is using a UAS to provide visual and 3D scene information for ground support in potentially unknown or altered environments. The "birds eye view" from the UAS can provide information quickly to a ground vehicle or ground support team to reduce response time in temporal critical missions. Additionally, the UAS can provide details of the environment that would otherwise be lost from ground sensing alone.

The benefit of aerial imagery is accentuated in cases of natural disasters where a priori information may no longer be relevant. In these environments, a UGV alone may have difficulty finding any path, let alone an optimal path. Therefore, a UAV can be used to provide greater high-level detail of the environment and explore a more efficient path.

1.2 Approach

The issue of global 3D reconstruction is trivial if perfect knowledge of the UAS pose is known, and there is no sensor/measurement noise. With knowledge of the true pose, and perfect

camera calibration, the 3D points can simply be projected into the world scene without the need for point cloud registration. However, only an estimate of the true pose is known in reality, and image sensors are subject to noise. Directly using a noisy pose estimate results in a discontinuous and or blurred scene. Therefore, to reconstruct a global scene, it is advantageous to use the global position estimate, but allow point cloud alignment through orientation refinement.

While 3D reconstruction algorithms exist already in the literature and are available in a variety of closed and open source implementations, many of these algorithms do not globally align the scene. Additionally, with a common use-case for stereo depth information applied to self-driving cars, the existing implementations rely on rich scene texture with both near and far points for pose estimation. Therefore, the need for globally aligned 3D reconstruction in low texture, low obstacle density environments, is an area of study in need of further development.

In this thesis, a framework for semi-dense 3D reconstruction using stereo cameras affixed to a UAV for the purpose of ground vehicle path planning is presented. The approach seeks to improve obstacle registration by fusing feature matched ICP estimation with the RTK-GPS constrained flight controller pose estimate.

It is important to emphasize that the reconstruction algorithm must fit within a frame-by-frame pipeline in order to reduce impact on the existing system. This constraint eliminates the possibility of performing global scene correction as is commonly done for most reconstruction frameworks.

Chapter 2

Review of Literature

2.1 Applications

Visual perception, including 3D reconstruction, has a broad range of applications. In fact, due to advancements in computer vision (including large open-source software libraries and better hardware), the usefulness of this technology is being explored in a variety of fields from mechanical and civil engineering, oceanography, and bio-medical to name a few. For example, in the space of civil engineering, 3D reconstruction is being used for structural health monitoring as well as building site management [14]. Additionally, Bergamasco et al. [3] presented a stereo reconstruction pipeline for scientific study of ocean waves, and Maier-Hein et al. [15] presents approaches to soft tissue geometry reconstruction for robotic assisted minimally invasive surgeries (Figure 2.1c). Other common applications for 3D reconstruction include geological surveying and mapping, and of course, robotic perception.

There is of course more than one approach to obtaining 3D data. Light Detection and Ranging (LiDAR), sonar, RGB camera images, and structured light or Time of Flight (ToF) RGB-D sensing such as from Kinect sensors, are a few of the common sensors used to obtain 3D data.

LiDAR estimates depth by calculating the time it takes for light to return to the sensor [18]. Overall, Integration of a LiDAR sensor can be simpler compared to a stereo camera sys-

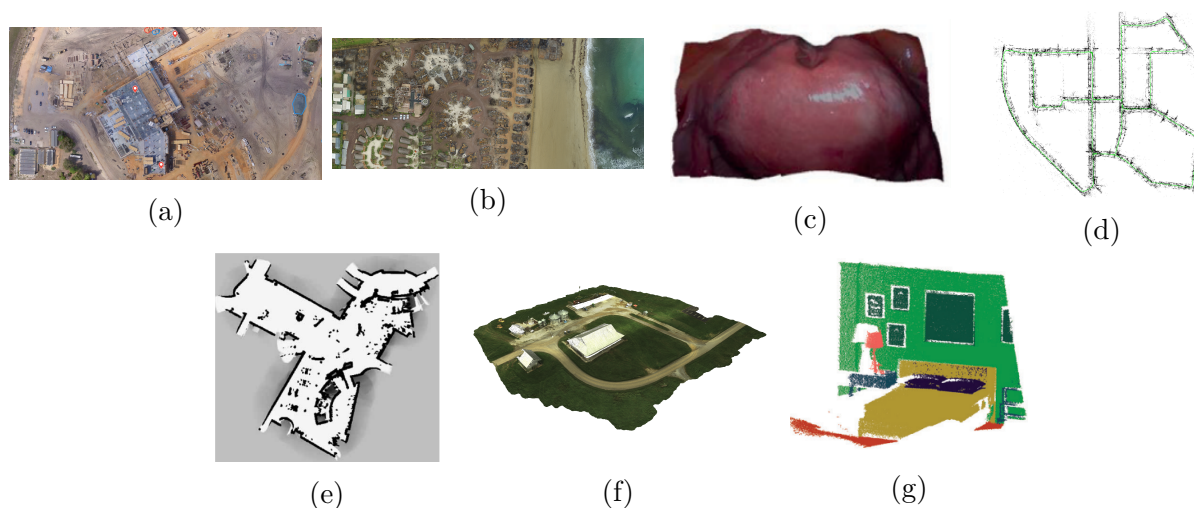


Figure 2.1: Applications of 3D reconstruction. Construction site management (a) and surveying (b), bio-medical (c), localization (d), mapping (e), modeling (f), segmentation (g). [7][15][17][24][23]

tem because the sensors do not require frequent calibration and complex post processing to produce a 3D point cloud. However, LiDAR data requires high bandwidth for transmission which can become problematic in online multi-robot and human in the loop applications. Additionally, the inclusion of moving mechanical parts, affordability of high-resolution devices, and lack of RGB information makes them less ideal in some cases. Recently, solid state LiDAR devices have been developed making the technology more robust, and in smaller form factor for easier integration [19].

RGB-D cameras use an infrared emitter-detector to determine depth information. This is generally accomplished in one of two ways: structured light, and time of flight. In the structured light approach, depth is estimated by the measured deformation of the projected light patten [28]. Similar to LiDAR, the ToF camera estimates depth based on the time it takes for the infrared light to return to the sensor. RGB-D cameras capture an RGB image as well as depth so that color information of the scene is also obtained. However, infrared based sensors can be severely limited in range (about 3 meters [22]) an have trouble

capturing some features in outdoor environments due to light sensitivity [18].

Compared to active sensing (LiDAR and RGB-D), passive sensors do not produce depth information directly. Instead, cameras can be used to obtain depth information by photogrammetric means. In the monocular case, 3D information is determined by some variant of structure from motion (SfM) photogrammetry [10]. However, monocular estimation can suffer from dramatic errors in drift due to scale ambiguity. Conversely, a stereo camera pair can remove the scale ambiguity and determine depth information frame by frame through stereo triangulation [17].

The net result of all such sensing techniques is rudimentary 3D information: 3D point cloud data. Popescu et al. [18] compared results obtained from these three main sensing techniques: Laser Ranging, photogrammetry, and infrared. Results show that there is more detail with laser scanning; however, photogrammetry provided more scene context with RGB information. Infrared provided less detail overall but could be sufficient in some applications. Figure 2.2 demonstrates 3D point clouds produced of the same scene using these different sensing techniques.



Figure 2.2: Comparison of laser, photogrammetry, and infrared sensing on two different scenes. Tunnel (top row) and bridge (bottom row). Laser ranging (left column), photogrammetry (middle column), infrared (right column) [18]

In the realm of robotic perception, 3D data is often used for localization and mapping. With the progression of autonomous car technology, dense 3D perception is important to ensure autonomous vehicles can safely interact with the environment [4]. While these mapping techniques allow autonomous systems to gain a general understanding of occupied space in the scene, segmentation and machine learning techniques can be applied to 3D data to add more detailed information about types of objects in the scene as well.

Tchapmi et al. [23] made significant contributions to the field of 3D scene segmentation in 2017 to improve robotic perception of the 3D environment. With greater scene understanding, autonomous systems can be designed to interact with the scene with improved intelligence as well as adapt to dynamic environments.

In this thesis, stereo cameras are used to reconstruct a 3D environment. Therefore, the following sections will touch on the literature associated with photogrammetry including feature matching and tracking. Additional background supporting computer vision are discussed in chapter 1.1.

2.2 Feature Matching

In computer vision, an image feature is a mathematical description of the uniqueness of a pixel or region of pixels. In general, the more unique the pixel is, the stronger the feature. Feature matching is really broken into three distinct steps: detection, description, and matching [1]. One of the oldest and most used feature detectors is the Harris Corner Detector presented in 1988 by Harris and Stephens [9]. This algorithm extracts image edges using gradient filters in order to then classify corners from the detected edges. While there are applications for such a descriptor, it is not invariant to scale, and therefore not useful in mapping applications where image scale is subject to change due to camera pose and noise.

While there are a handful of scale and rotation invariant feature descriptors presented in the literature, two of the more widely adopted versions suitable for online use are ORB, and SURF.

2.2.1 ORB

Oriented FAST and rotated BRIEF (ORB) features are a common choice when looking for fast feature descriptor matching. As the name implies, ORB is a combination of the FAST keypoint detector and BRIEF descriptor [20]. ORB has readily available OpenCV libraries making it easy to implement in C++ where speed for online applications is important. ORB has been shown to outperform Scale-Invariant Feature Transform (SIFT) processing time by as much as 80%; However, compared to SURF, ORB showed comparable performance in terms processing time and match rate [12].

2.2.2 SURF

Speeded Up Robust Features (SURF) are also robust to rotation and scale. SURF is built around the "Hessian matrix-based detector and distribution-based descriptor" [1]. The Hessian matrix is defined about a point p , and scale, σ , and is given by

$$H(p, \sigma) = \begin{bmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{yx}(p, \sigma) & L_{yy}(p, \sigma) \end{bmatrix} \quad (2.1)$$

where, the matrix is composed of second order Gaussian derivative convolutions of the image in point p .

In SURF feature detection, a 9×9 discrete approximation to the Gaussian derivative is used to detect blob regions in the image. Figure 2.3 demonstrates the progression of approximating

the Gaussian derivative as presented by Bay et al. [1].

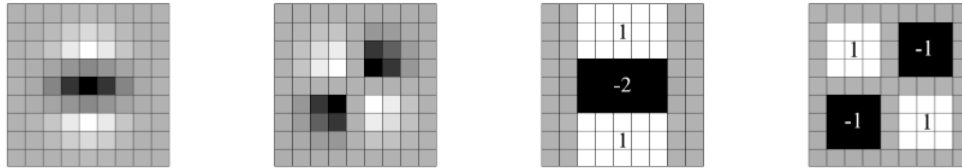


Figure 2.3: The Gaussian derivative is discretized into a 9x9 grid and approximated. The two images on the right depict the partial derivative in y - and xy -directions respectively [1]

SURF features have been shown to be a computationally inexpensive approach to robust feature matching. Additionally, there are MATLAB libraries which support SURF feature extraction and matching for ease of use. Since the performance between ORB and SURF are comparable [12], SURF features were chosen for this thesis because of the integration with MATLAB.

2.3 SLAM

Simultaneous Localization And Mapping (SLAM), is the process of both mapping based on sensor measurements, as well as localizing a vehicle within the map. While this thesis does not present a SLAM implementation, there are notable open source SLAM packages worth mentioning because they inherently produce a sparse 3D map. ORB-SLAM, and LSD-SLAM are two packages available with ROS integration which make them relatively easy to use when getting started with stereo SLAM. As with most 3D SLAM methods, these implementations rely on landmark detection and registration.

2.3.1 ORB-SLAM

Some online SLAM implementations by design create and update a sparse reconstruction of feature points (landmarks). A common open source implementation for visual SLAM with stereo support is ORB-SLAM2 [17]. ORB-SLAM2, as the name implies, uses ORB features extracted from images to estimate pose. However, the algorithm relies on both near and far points to properly estimate pose. Without far points (points where disparity is below a specified threshold), ORB-SLAM may not resolve orientation well and the 3D map accumulates errors. Therefore, ORB-SLAM works well in applications such as autonomous driving because of the richness in visual depth/texture, but is not as capable in low texture applications.

2.3.2 LSD-SLAM

LSD-SLAM functions similarly to ORB-SLAM, but in [6], the algorithm has already been designed to fuse odometric data to aid in feature match outlier removal. Typical for any visual odometry methods, both ORB-SLAM and LSD-SLAM implementations suffer from accumulated error and drift. Without loop closure or additional filtering to correct pose, (i.e. Kalman filtering and sensor fusion using GPS, odometry, etc.) these algorithms may not produce a low error reconstruction overall. Additionally, without fusing data from a global positioning system, it is not possible to provide a globally aligned map and the results are restricted to a local map.

2.4 Implementations

As there are more than one approach to obtaining 3D data, there are also more than one way to store that data in a reconstructed 3D scene. One common approach is to process the 3D data into a probabilistic occupancy grid. In this way, a voxelized map is used to represent either occupied or unoccupied voxel in the scene. Hornung et al. [11] provided an open source framework for 3D mapping based on octrees. Figure 2.4 [11], shows an example set of point cloud data being used to generate an octmap.

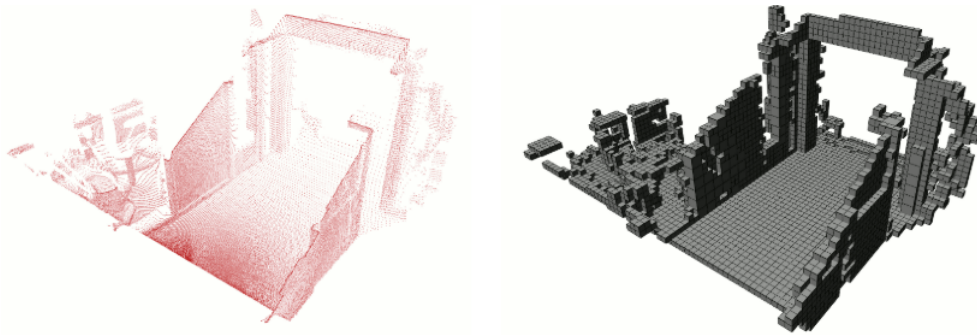


Figure 2.4: Point cloud data (left), is used to generate an octree map (right) showing occupied voxels. [11]

Such an approach is robust to noisy sensor measurements, but discretizing the map into voxels means that some information is lost due to resolution. Despite this inherent downside, such mapping techniques are typically sufficient for robot navigation in many situations. Ultimately, all robotic mapping implementations require some amount of space discretization as they operate in the discrete domain.

However, dense 3D reconstruction can provide additional information about the scene that may be useful for human-in-the-loop systems, or segmentation and robotic interpretation. Dense reconstruction approaches often rely on visual odometry and sparse feature matching as proposed by Geiger et al. [8].

Milella and Reina [16], presented a multi-baseline stereo approach to 3D scene mapping for autonomous driving. The multi-baseline system provides the ability to detect obstacles both near, and far from the vehicle. Additionally, Mur-Artal and Tardos [17] provides an open source Simultaneous Localization and Mapping (SLAM) implementation which supports stereo camera sensors for 3D mapping and navigation.

Kumar [13] developed a stereo 3D reconstruction algorithm for the same application described in this thesis. Similar to the approach that will be presented in Chapter 4, the reconstruction algorithm relies on sparse feature matched point cloud alignment using ICP to estimate the frame-to-frame camera pose. An additional step includes a frame-to-model match using images within a region of interest to avoid accumulating errors. As demonstrated in Figure 2.5 [13], the implementation produced a clean reconstruction which was well aligned locally; however, the results lacked global alignment. Additionally, the implementation was not prepared for on-board integration with the existing ROS-based system, and required storing all images in memory (necessary for frame-to-model matching) which is not practical for large map building.

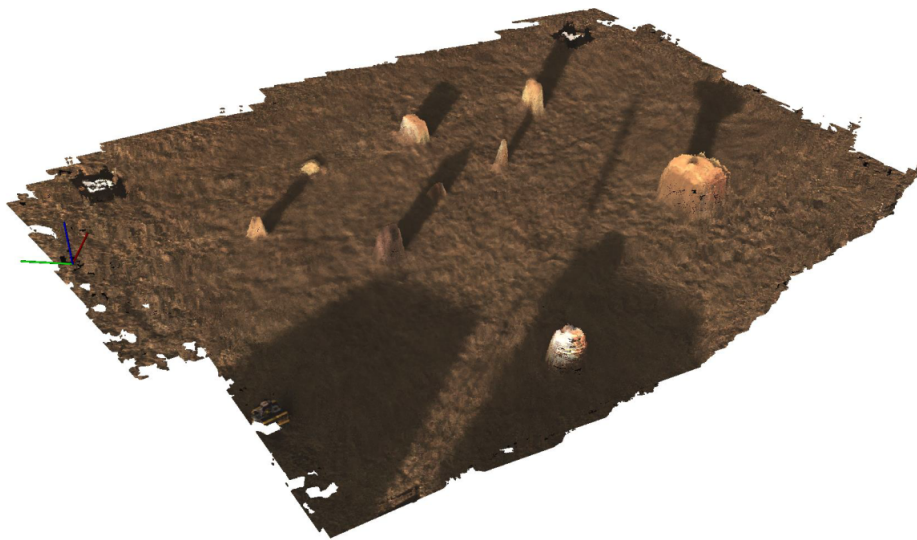


Figure 2.5: Dense 3D reconstruction from aerial stereo imagery. [13]

There are a few Off-line Commercial Off The Shelf (COTS) sequential image reconstruction programs which can produce exceptional results, but at the cost of long computation time. Agisoft is a photogrammetry program which uses a sequential monocular images to estimate camera poses and reconstructs a 3D scene using global optimization. DroneDeploy is similar program which can be used to produce 3D maps and is specifically targeted for surveying applications with camera equipped UAVs. However, such off-line programs still require GPS surveyed points (Ground Control Points) to aid in reconstruction scale and global positioning. Additionally, for many surveying applications, speed is not a priority. Therefore, these particular commercial products can produce quality reconstruction results, but can take hours or days to process.

Chapter 3

Background

This work is driven by the need for global, online 3D stereo reconstruction from aerial imagery, deployed on a system that is easy for future researchers to build off. The purpose of the reconstruction is predominantly for use in obstacle detection for ground vehicle path planning. This chapter presents the detailed motivation behind this thesis, background theory related to stereo vision, hardware design, and the test environment.

3.1 Project Motivation

Unmanned ground vehicles (UGVs) are becoming increasingly popular in a variety of deployment applications from military to search and rescue. UGVs can provide cooperative support such as hauling, debris removal, and tactical support. Often, an unmanned ground vehicle is deployed without a priori information about the environment, or to locations where the environment may have been altered in some way (i.e. after a natural disaster). In these instances, it can be difficult for a UGV alone to efficiently find a path through the environment to a pre-defined goal. Therefore, it is the purpose of this project on the whole to develop a collaborative UAV/UGV system to find an optimal path for the UGV. The UAV, equipped with a custom downward facing stereo camera, provides high level information about the 3D environment so that a path for the UGV can be determined.

The existing robots at the Unmanned Systems Lab have already been under development

and contain much of the communication systems, path planning and navigation, as well as basic stereo processing which is used to produce a 2D cost map. However, the 3D data is projected into the scene using only the raw UAV pose estimate provided by the Flight Control Unit (FCU). This pose estimate is constrained well in position with the inclusion of RTK GPS. However, orientation is noisy which leads to blurring/dilation of obstacles as the same object may be projected into different locations in the scene frame to frame.

Previous work in this project aimed at creating a well aligned 3D reconstruction for the purpose of improved obstacle mapping, have been foiled by lack of integration with the existing ROS system. Additionally, previous attempts were not designed in a manner that supports online operation. While such work produced quality local maps fast enough for online use, there was not a good path forward to real system integration.

The primary goal of this thesis is twofold: improve the global registration of obstacles in the scene and develop the algorithm in a framework conducive to ROS integration and online operation. An additional soft requirement of the project is to develop within a framework that is easy for new researchers to understand and continue to develop with as the project grows.

3.1.1 Obstacle Detection

While there are several applications for online 3D reconstruction as discussed in section 2.1, the original motivation of this thesis requires that 3D information be used for obstacle detection. With depth information provided by the stereo cameras, it is possible to determine the relative altitude of points in the 3D scene. To reduce computation, 3D data is first averaged into grid cells with a user defined grid size. The gradients between cells are then computed to produce a 2D cost map of the environment. Obstacles are naturally defined as

areas with large gradients (lethal cost).

The obstacle map produced by the UAV is used in collaboration with a UGV. Therefore, in this configuration, the UAV acts as the exploration vehicle. To explore, A UGV goal position is set by the ground control station (operator). The UAV will then search for a path the UGV can follow by flying over the scene and producing an obstacle map.

3.2 Computer Vision

The field of computer vision has been a popular research topic for the past several decades. While there is sufficient work detailing projective geometry, camera matrices, and stereo vision, this section provides an overview of the basic concepts and theory necessary for stereo vision leading to 3D reconstruction.

3.2.1 Camera Model

A common approach to mathematical camera modeling, is by way of the "pin-hole" camera model and projective geometry. In this approach, the camera's optical center is modeled as a simple point in space. Figure 3.1 demonstrates how the pin-hole camera model is used to map a 3D point $(X, Y, Z)^T$ to the image plane [10].

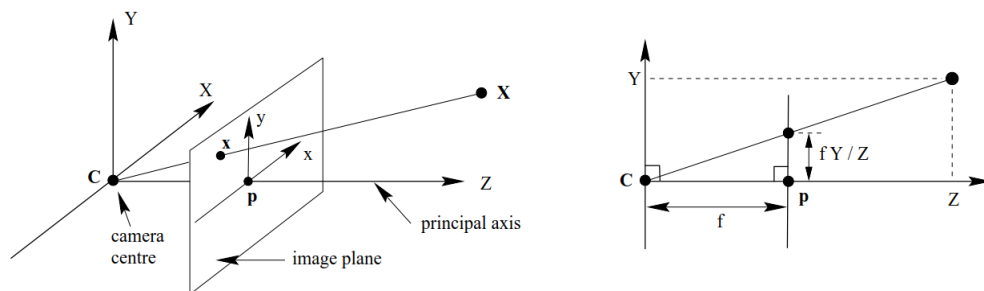


Figure 3.1: Pinhole camera Model [10]

In general, a homogeneous world point, \mathbf{X} , can be transformed to an image point, \mathbf{x} , through means of the camera projection matrix as

$$\mathbf{x} = P\mathbf{X} \quad (3.1)$$

where \mathbf{x} is the 3-element vector $(fX, fY, Z)^T$, \mathbf{X} is the 4-element homogeneous world point $(X, Y, Z, 1)^T$, and P is the projection matrix given by

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (3.2)$$

where f , is the focal length. A more general mapping from the world point to an image point must account for the offset from the principle point (optical center). The camera calibration matrix (also known as the intrinsic matrix), K , accounts for this offset by including the principle point as shown in Equation 3.3

$$K = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \quad (3.3)$$

where (c_x, c_y) is the principle point in pixels. From Equations 3.1 and 3.3, a modified form of the projected point is given by

$$\mathbf{x} = K[I|0]\mathbf{X} \quad (3.4)$$

Finally, to account for rotation and translation from the camera frame to the world frame, it is necessary to modify the projection matrix slightly. The orthogonal rotation matrix, R , and the translation vector, T , (together known as the camera extrinsic parameters) simply

replace the identity matrix and zero vector in Equation 3.4 to form

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K [R|T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (3.5)$$

where, in this formulation x and y are the image pixel coordinates, and $(X, Y, Z, 1)^T$ is the homogeneous world point coordinate.

3.2.2 Stereo Vision

As discussed in Section 3.2.1, it is possible to relate a 3D world point to the 2D image plane with a single image sensor, but depth information is lost. However, the distance a 3D point lies from the camera can be obtained using two sensors, pixel correspondence, and triangulation. Stereo Vision can essentially be summarized as the process of using pixel correspondence between two image frames (i.e. a left-right stereo image pair), in order to determine the distance (disparity) between each corresponding image pixel between the left and right image. This disparity can then be related to depth directly by

$$Z = \frac{B \times f}{d} \quad (3.6)$$

where, Z is the depth in meters, B is the distance between image centers in meters, f is the focal length of the image sensor, and d is the disparity.

However, with modern camera resolution on the order of 10^6 pixels, determining pixel correspondence across the entire image is unconscionable. Therefore, stereo rectification is performed to reduce the complexity in disparity computation.

Stereo rectification is the process of aligning the left and right image frames such that the epipolar planes are aligned. This so called "epipolar constraint" reduces the problem of point correspondence to a one dimensional search rather than globally. Figure 3.2 demonstrates how epipolar geometry can be used to simplify pixel correspondence. Additionally, Figure 3.2 demonstrates why depth cannot be extracted from a single sensor alone as the pinhole camera model is ambiguous along the line $\bar{C}X$.

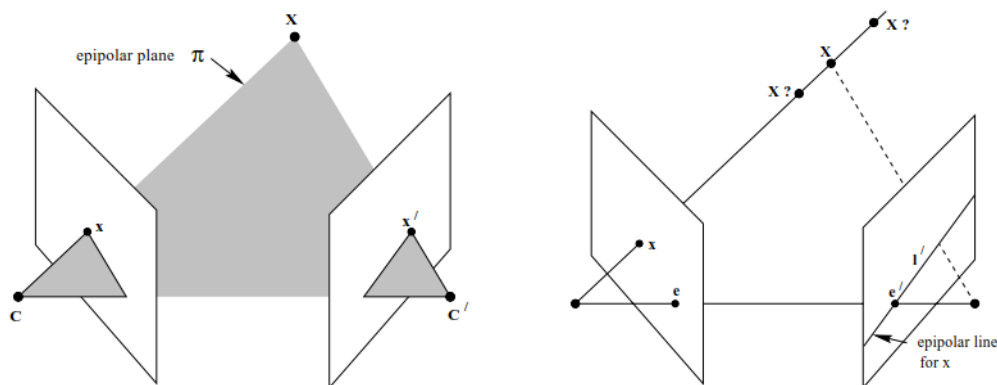


Figure 3.2: Epipolar geometry reduces correspondence complexity to 1D search along epipolar lines. [10]

Given the significance of the epipolar constraint, stereo rectification has a significant impact on the quality of stereo reconstruction. In addition to determining rectification parameters for the left and right images, stereo calibration also determines camera lens distortion coefficients used to remove effects of lens distortion on the images.

In this project, the ROS camera calibration package was used to calibrate the stereo cameras. The ROS calibration is relatively easy to use and provides live feedback of the calibration process. Figure 3.3, demonstrates the calibrator interface and the process involved in calibration. The procedure requires that the predefined checkerboard be swept in front of the cameras at various distances and skew angles in order to adequately estimate the intrinsic and extrinsic camera parameters.

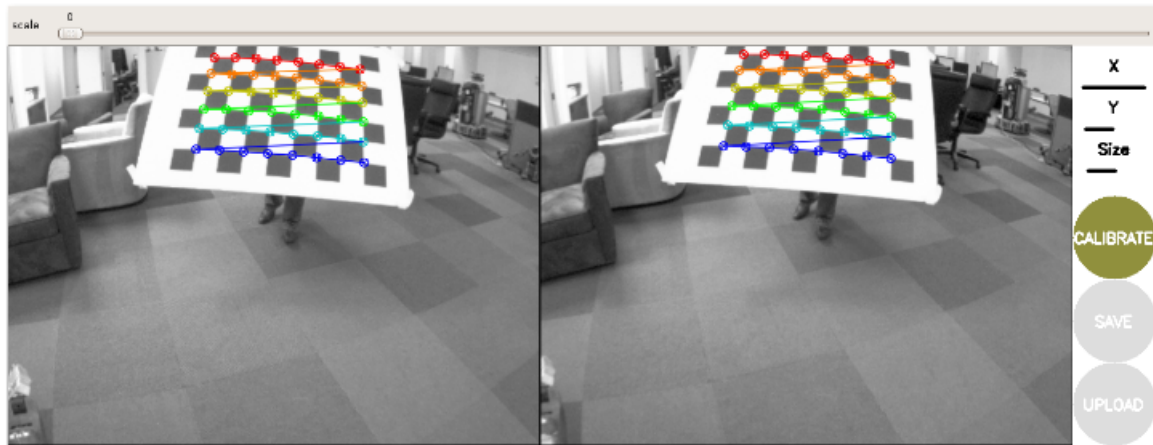


Figure 3.3: ROS camera calibration interface. [26]

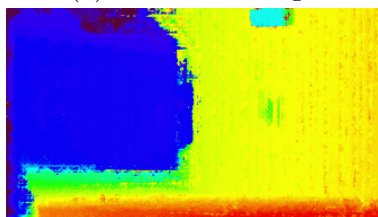
Once the calibration is complete, the camera parameters can be used with the stereo semi-global matching algorithm to produce disparity maps. Figure 3.4, shows reasonable disparity images produced following camera calibration.



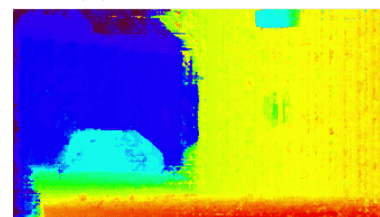
(a) Raw cam image



(b) Raw cam image



(c) Disparity



(d) Disparity

Figure 3.4: Disparity after camera calibration shows a car passing through the scene

3.3 Coordinate Frames

Initially, 3D points are projected following the procedure for re-projecting points discussed in section 3.2.2. As such, the 3D point reconstruction is initially in the camera coordinate frame. However, since the goal is to obtain a global scene, it is necessary to transform the 3D points from the camera frame to the world frame.

The camera frame follows the typical convention found in the state of the art where z is positive out of the image plane, x is positive for increasing image pixel column, and y is positive for increasing image pixel row. Figure 3.5. further illustrates the camera frame convention.

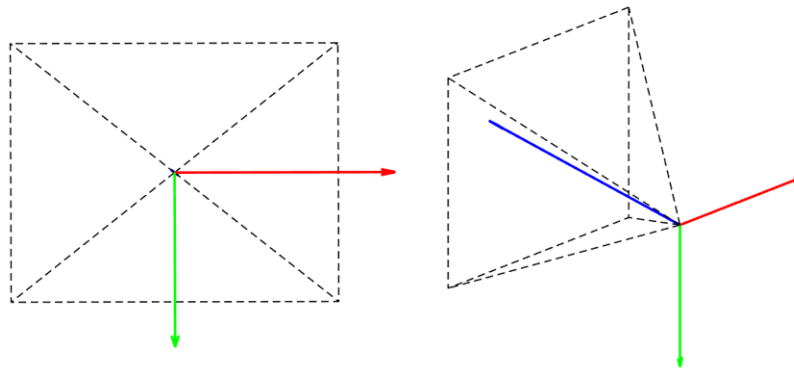


Figure 3.5: Camera coordinate frame convention is x positive along image right, y positive along image down, and z positive out of camera.

The camera frame must also be related to the UAV's `base_link` pose. The `base_link` pose of the UAV is the frame which flight controller global pose information is provided. Figure 3.6 demonstrates the relationship between the camera frame to the UAV `base_link` frame. The process to represent points in the camera frame with respect to the `base_link` frame is carried out following methods of homogeneous transformations.

The homogeneous transformation matrix is used to relate a homogeneous point in one frame to the corresponding coordinates in another frame. In this case, the points associated with

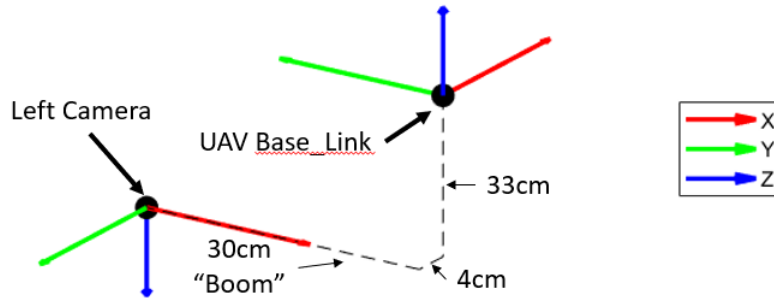


Figure 3.6: Coordinate frames for the camera and the UAV base link, demonstrate translation and rotational transformation required to interpret the camera frame in terms of the UAV and (in turn) global frames.

the camera frame are transformed to world coordinates. The homogeneous transformation matrix to transform coordinates in frame 1 to frame 2 follows the form

$$T_{12} = \left[\begin{array}{c|c} R_{12} & P_{12} \\ \hline 0 & 1 \end{array} \right] \quad (3.7)$$

where, R_{12} , is the 3x3 orthogonal rotation matrix representing the rotation from frame 1 to 2, and P_{12} , is the 3x1 vector representing the location of point P with respect to frame 2 [2].

the rotation component of the transformation matrix can be expressed as the multiplicative combination of basic rotations about a given axis,

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\theta & \cos\gamma \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \quad R_z(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where, R_x , R_y , and R_z are the rotations about the x-, y-, and z-directions respectively.

The order of basic rotations is important and there is more than one formulation that can

produce the same result. In this thesis, the conventional ZYX order is adopted. From Figure 3.6, the required rotations to relate a point in the camera frame to a point in the UAV base_link frame are 90 degrees about the z-axis, followed by 180 degrees about the y-axis,

$$R_{bc} = R_z(90)R_y(180)R_x(0) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

where, R_{bc} represents the rotation from the camera frame to the UAV base link.

Therefore, with the translation vector between the camera and the base_link (with respect to the base-link frame) as defined in Figure 3.6, the transformation matrix from the camera frame to the base_link frame is given by

$$T_{cb} = \begin{bmatrix} 0 & -1 & 0 & -0.04 \\ -1 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & -0.33 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.9)$$

As a property of transforms, the inverse of the transformation defined in Equation 3.9 is also used to define points in the base_link frame to the camera frame.

The transformation above is used to define the points in the camera frame to the UAV base_link frame, however, the 3D points must still be transformed to world coordinates. By following the same process above, the transformation from the base link to the robot origin in the global frame can be defined as

$$T_{bw} = \left[\begin{array}{c|c} R_{bw} & P_{bw} \\ \hline 0 & 1 \end{array} \right] \quad (3.10)$$

where, R_{bw} , represents the UAV orientation in the global frame and, P_{bw} , is the location of

the `base_link` with respect to the origin.

Finally, the complete transformation of a point in the camera frame, to a point in the world frame is given by multiplying the transformation between the camera and `base_link`, by the transformation between the `base_link` and the world frame

$$T_{cw} = T_{cb} \times T_{bw} \quad (3.11)$$

3.4 Mechanical Design

This section presents the mechanical design improvements made to the prototype stereo camera boom developed as part of the work completed by Kumar [13]. Specific improvements made to the system include sustainable calibration by way of non-conforming aluminum mounts, hexagonal boom to improve repeat-ability in re-mounting, and precise adjustment to camera alignment using setscrews. In this section, the original prototype is discussed, along with the improvements made to the system.

3.4.1 Prototype

The prototype stereo boom was designed and integrated by Kumar [13]. The prototype was originally designed for potentially extra-wide baseline measurements. Figure 3.7 shows the long boom attached to the UAS was largely unused as only a 60 cm baseline was used during testing.

Figure 3.8 provides a closer view of the prototype camera mounts which were made of 3D printed ABS plastic. The ABS was subject to deformation due to temperature changes. Additionally, the mounts did not provide reliable stiffness and the friction fit against the



Figure 3.7: Prototype mounted to UAS

round tube made alignment difficult. As a result, the calibration procedure often took a lot of time, and calibration would not last transportation or temperature changes.

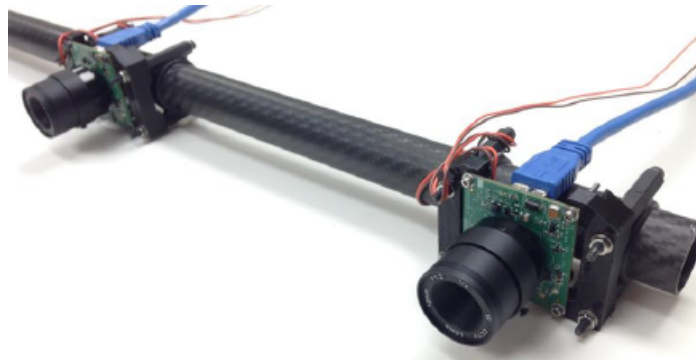


Figure 3.8: prototype used 3D printed ABS for camera mounts on a round carbon fiber tube

While the prototype served its purpose of proving the technology works, the design needed to be improved to make the process more repeatable. Additionally, since stereo camera calibration can be a time-consuming process, the need for a more rigid design less susceptible to variation in temperature and transportation was required.

3.4.2 Design

As discussed in Section 3.4.1, improvements to the mechanical design were needed in order to maintain calibration for longer periods of time, as well as simplify the adjustment (camera alignment) and installation process. In this section, the new design is presented using hexagonal carbon fiber tubing, and aluminum mounting brackets. Figure 3.9 shows the complete re-designed stereo camera assembly.



Figure 3.9: New stereo camera boom is easier to align and maintains calibration longer.

Mounting

Stereo cameras need to maintain relative alignment after they have been calibrated in order to continue functioning properly. If the extrinsic properties are altered due to mount deformation, then proper point correspondence will not be possible, and the disparity results will be incorrect. Therefore, new aluminum camera mounts (Figure 3.10) were designed to replace the old 3D printed mounts. The new mounts provide greater consistency in mounting and maintain calibration longer.

Since the new camera boom has a hexagonal cross section, installation of the camera mounts, as well as mounting the boom to the UAV, is more repeatable in terms of mechanical orientation. Additionally, the excess length of tubing has been removed in the new design.

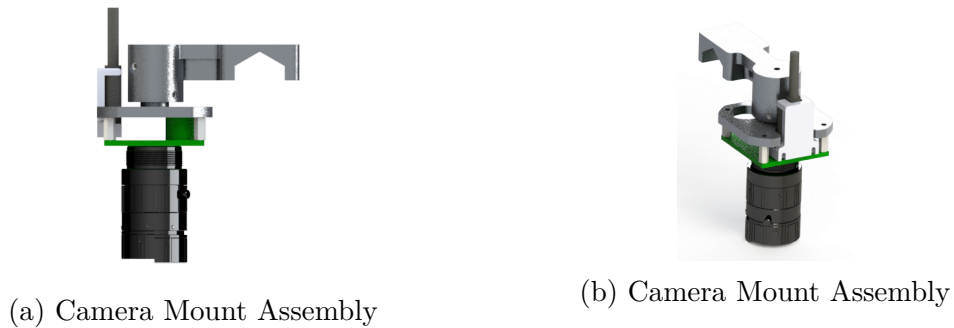


Figure 3.10: Camera Mounting

Alignment

As discussed in section 3.2.2, stereo rectification is performed as part of the calibration process to align the camera epipolar lines. However, to improve calibration results and avoid over-rectification of the images, good initial alignment of the cameras is necessary. The new design allows for both Planer alignment of the cameras by adjusting setscrews, as well as rotational adjustment achieved by loosening the camera mount screws and rotating the camera.

Figure 3.11 shows a detailed view of the adjustment features. Specific features include ball point setscrews and a dimpled shaft to allow adjustment of the camera mount.

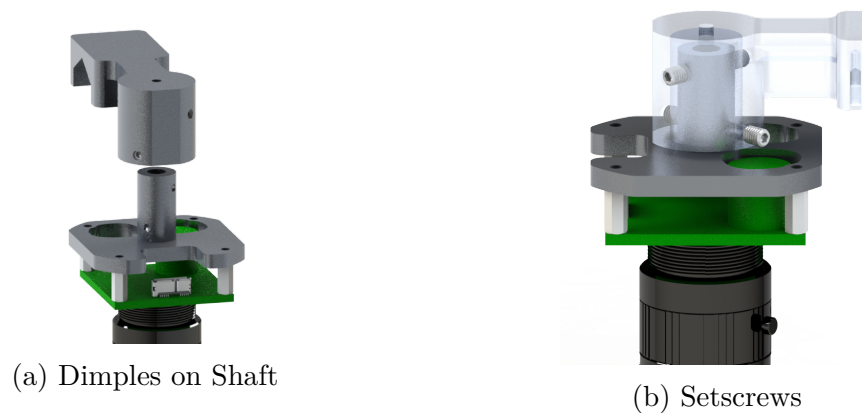


Figure 3.11: Planer alignment achieved with setscrews

3.5 Environment

In this section, the simulated and physical environments are presented.

3.5.1 V-REP

The Virtual Robotics and Experimentation Platform (V-REP) by Coppelia Robotics, was used to produce the simulation environment. In addition to an easy to use interface, V-REP has a MATLAB api which provides simulations control and data collection directly through MATLAB. Since MATLAB was the primary development environment used throughout this project, this ease of interfacing the simulation environment with MATLAB was a benefit. Compared to other common robotics simulators, such as Gazebo, V-REP was fairly simple to understand and deploy for the purposes of this project.

The simulation design was fairly simple as the purpose was to test the 3D reconstruction algorithm. Therefore, only a few simple objects were added to a plain checkerboard scene floor.

The stereo boom was simulated to represent the actual boom. The stereo baseline was configured to 60 cm. The cameras were configured with 30 deg. FOV and 1280x720 resolution to match the physical cameras.

Simulation conditions were chosen to relate to real world test conditions. Therefore, the stereo cameras were controlled at an altitude of 22 meters while capturing images every meter of traversed distance. The movement was simple frame to frame translation of 1 meter along the global x-direction (along the length of the scene) as defined in in Figure [3.12](#).

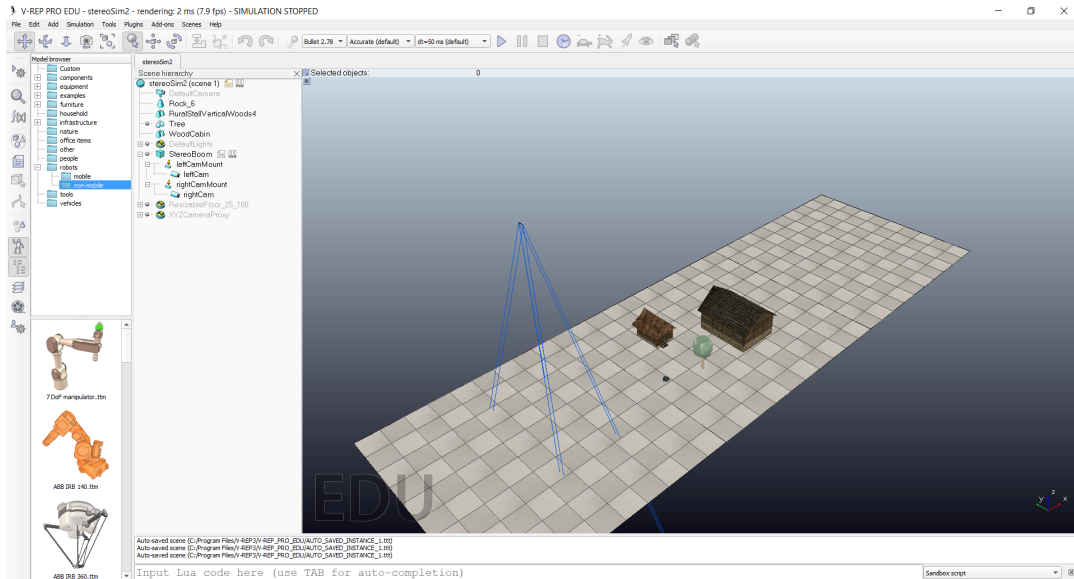


Figure 3.12: V-REP Environment

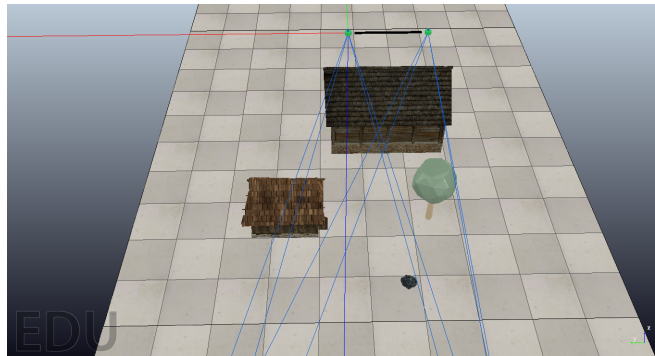


Figure 3.13: Close-up of simulated stereo boom and V-REP scene

3.5.2 MATLAB

Frequent turnover of graduate researchers in the unmanned systems lab, and any university for that matter, is typical and usually results in the loss of information when the researcher completes his/her degree. Ideally, contributions would be well documented and integrated; however, often the work is incomplete, and it can be difficult or impossible for someone new to continue the work without spending significant time learning and understanding what has already been done. Therefore, additional factors motivating this project is platform develop

to allow new researchers to quickly understand and modify the algorithms. To this end, MATLAB was chosen as the primary development platform for the 3D reconstruction.

MATLAB provides numerous toolboxes and excellent built in visualization tools, making it an ideal prototyping platform for experimentation and development. In the following sections, the packages of greatest significance for this project are briefly discussed.

Computer Vision Toolbox

MATLAB's computer vision toolbox provides functions for image filtering as well as a point cloud library, which is an essential tool for this project. Additionally, while it was possible to incorporate the ROS camera calibration results directly, it was often easier to use the camera calibrator app within MATLAB's computer vision toolbox. In this way, the images saved from the ROS calibration are simply loaded into MATLAB's calibration tool to obtain the stereo camera parameters directly. This process saves some computation time on initialization as well as eliminates scale errors which would sometimes be introduced from using ROS calibration parameters directly.

Robotics System Toolbox

MATLAB's robotic system toolbox was used in this project primarily for ROS interfacing. With the robotic system toolbox, MATLAB is able to communicate with a ROS master as a node just as any other device. This is useful for collecting ROS messages for visualization and online testing of the developed algorithm, as well as testing existing ros packages against the data being processed in MATLAB.

A significant shortcoming in this area, is the lack of message synchronization in MATLAB's robotics system toolbox. In this thesis, it is necessary to synchronize both left and right

camera images, as well as the UAV position data who processing. Therefore, a simple synchronizer is defined in section 4.3.2 to address this issue.

C-Coder

While not specifically explored in this thesis, MATLAB also provides a c-coder toolbox designed to convert MATLAB scripts to C or C++ code. For this to work well, the MATLAB script must be written in a C friendly way (i.e. mindful of memory allocation, data types, classes, etc.) to avoid significant errors on compilation. The C-Coder toolbox is intended to allow for high performance deployment of a MATLAB program once the prototyping stage is complete.

Visualization

Finally, MATLAB's abundance of built in visualization tools makes it an ideal platform for testing and refinement of the underlying reconstruction algorithm. With the ability to easily visualize images, feature matching, point clouds, pose graphs, etc... exploring how changes to the reconstruction algorithm impact the overall quality of the results is a relatively painless process.

3.5.3 Test Conditions

To fully evaluate the performance of the reconstruction presented in this thesis, real data is collected for offline processing as well. Real world data was collected at the Kentland Experimental Aerial Systems (KEAS) lab. Figure 3.14 demonstrates the test flight plan produced using mission planner. A flight pattern was chosen to include multiple passes that would overlap to test tracking and multi-pass registration. This "lawn-mower" pattern

is programmed with an altitude of 26 meters, ground speed of 3 m/s, and spacing between passes of 2 meters to provide a target image overlap of about 80-90%. Additionally, the scene included objects that varied in size from as small as the RTK GPS antenna, to a large barn. The scene also includes an area of low texture over grass and gravel to test performance in varying degrees of scene richness.



Figure 3.14: Lawn-Mower pattern flight plan to collect real data for algorithm testing

Data is monitored on the ground station; however, all the flight control, image capture, and data logging is handled onboard the UAV. Image and position Data are stored in a binary bag file directly using ROS. The data is post processed in two ways: simulated online, and offline. In the simulated online scenario, the bag file is played back in the ROS environment while the reconstruction program developed in this thesis subscribes to the topics needed. Running the program in this way simulates how it would work online where it would process data as it is received. In the offline case, synchronized message data is extracted from the bag, and the image and pose data is read directly from disk for processing.

The hardware specifications for the experimental data collection are tabulated in Table 3.1 for reference. With the on-board TX2, the system is able to achieve about a 3-Hz frame-rate for stereo image processing (disparity computation and point cloud generation) when GPU acceleration is employed. The off-board computer is used to develop and test feature matched reconstruction performance as well as online ROS integration by running the off-board computer in parallel with the system during testing. However, the off-board computer is only able to achieve a framerate of about 1-1.3 Hz depending on filter settings.

Table 3.1: Test hardware specifications

Equipment	Description	Specification
UAV	Tarot 960 Hex Frame	max weight w/payload: 22 lbs
Onboard Computer	NVIDIA Jetson TX2	CPU: Quad-Core 1.73 GHz GPU: 256-core Pascal 1300 MHz RAM: 8GB 128-bit 1866 Mhz
Cameras	FLIR BFLY-PGE-31S4C-C	Interface: GigE Megapixels: 3.2 MP MAX RES: 2048 × 1536 Chroma: Color
Localization	RTK GPS	Nominal Accuracy: 1-cm horizontally 2-cm vertically
off-board Computer	Dell Precision 5200	CPU: 8-Core i7-7820HQ 2.9 GHz GPU: NVIDIA Quadro M1200 RAM: 32GB

Chapter 4

Methodology

In this chapter, the methodology behind the 3D reconstruction implementation is presented including an overview of the reconstruction algorithm, architecture, feature matched point cloud registration, and filtering.

Throughout this thesis, the term "frame" is used often to mean the state of the UAV. The state of the UAV, as relevant to this thesis, includes the physical state (global pose), as well as the state of the image sensors (left and right images). As such, the "previous frame" will refer to the state of the robot (left image, right image, pose) at the previous time step. Similarly, the "current frame" will refer to the state of the robot at the current time step.

4.1 Integration

To avoid potential confusion, it is necessary to outline some important concepts regarding the integration strategy for the work being presented. As already mention in section 3.1, a system already existed which included much of the stereo processing to be handled on-board the UAV. However, to lay the groundwork for algorithm development on another platform, much of the fundamental stages have been reproduced.

Figure 4.1 illustrates how some processes are duplicated between the on-board and off-board computers. This was done so that a complete development platform could be used for offline

processing. As shown within the dotted line in Figure 4.1, the first initial processing of images (computing the disparity and generating 3D points) is the same in both cases.

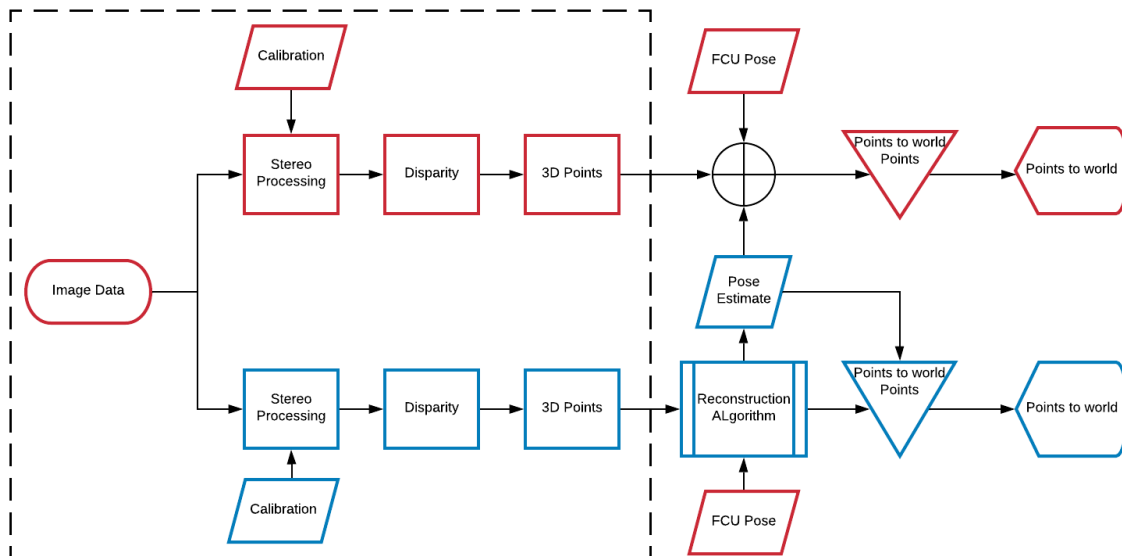


Figure 4.1: Part of the processing performed and development off-board, (blue) perform in parallel to what is done already on-board (red) the UAV during operation as illustrated by the similarity of processes within the dotted lines.

For online use in the field, the off-board reconstruction computer can be used to process the data and publish a camera pose based on the reconstruction algorithm. In this way, the FCU pose normally used on-board the UAV for 3D point re-projection, is replaced with this off-board pose estimate. The software framework already exists on-board the UAV to accept this "external pose estimate" as an alternative to the raw FCU pose. It is in this way that the reconstruction algorithm can be integrated into the existing robotic system to improve obstacle registration online. Additional off-board results include the reconstructed scene which can be visualized separately from the rest of the system; however, off-board processing speed could be increased if visualization of the full 3D scene is not necessary.

4.2 Reconstruction Algorithm Overview

In this section, a high-level overview of the reconstruction algorithm is presented. Specific details of each stage in the algorithm are discussed in the following sections of this chapter.

The reconstruction algorithm follows the basic process flow as illustrated in Figure 4.2. The primary processes that make up the algorithm are broken into 6 blocks: feature matching, outlier filtering, stereo processing (disparity and 3D point cloud generation from stereo parameters), ICP, Kalman filtering, and finally adding the current frame 3D point cloud to the scene.

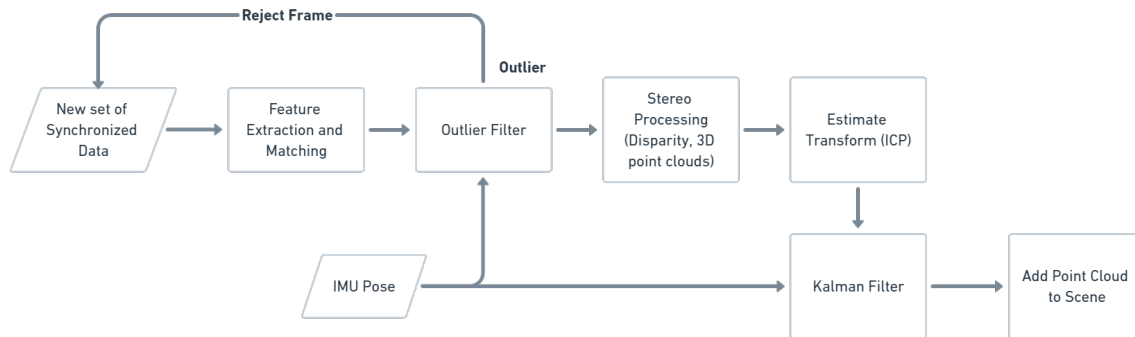


Figure 4.2: 3D Reconstruction algorithm illustrated as six primary block processes

As Illustrated in Figure 4.2, for each set of synchronized images and pose data, the first process is feature extraction. As will be discussed in Section 2.2 in more detail, this process involves extracting 2D SURF features from the grayscale left camera image for both the current and previous frames. Features between these frames are then matched using the Sum of Squared Differences (SSD) matching metric.

An additional outlier filter follows the initial feature matching/extraction process. The outlier filter serves two purposes: remove additional erroneous feature matches that may be left over after the previous step, and reject the frame altogether if there are insufficient features

for ICP registration or if there is a significant mismatch between the IMU measurement and estimated feature translation.

Once good feature matches have passed through the outlier filter, the stereo processing is performed. In this step, the disparity is computed from the left and right images, and both a semi-dense point cloud of the current frame as well as the set of sparse feature clouds are generated. The semi-dense cloud represents all the points that will eventually be added to the scene in the final stage of the reconstruction. The cloud is described as "semi-dense" because of down-sampling performed to reduce computation time. The sparse feature clouds represent the 3D point clouds of the current left image and the previous left image and are simply the re-projection of the matched 2D features into 3D points.

ICP (Iterative Closest Point) estimates the rigid body transformation between the sparse feature clouds of the current and previous frame. The transformation is used to estimate a current camera pose. The camera pose is then fused with the IMU pose in the Kalman filter process. Finally, the 3D point cloud is added to the accumulated scene using the pose output from the Kalman filter.

4.3 Software

The deployed robotic system runs on a Linux OS with ROS Kinetic, and most of the software integrated throughout the robotic system is written in either C or C++. These languages are great for speed but have a steep learning curve for some researchers who are hoping to contribute to the system. Additionally, C and C++ do not provide easily accessible visualization tools, and as a compiled language, are not convenient for prototyping/testing. Therefore, this project was built in MATLAB which is a great prototyping platform familiar to many in the engineering research and computer vision field.

4.3.1 Existing Code

The reconstruction algorithm is modeled in part after prior work intended for the same system in [13]. The prior reconstruction was developed in C++, but was only developed for offline use (i.e. no ROS integration or plug-and-play interface), and did not provide a straightforward path to system integration. While some of the underlying reconstruction framework has been borrowed from the previous implementation, the software overall has been completely re-developed in MATLAB.

As mentioned in Section 3.1, much of the robotic software for the OCS, UAV and UGV had already been developed. This included communications between the platforms, controls/-navigation, and stereo vision. The contributions made in this thesis are intended to be "plug and play" with the existing framework provided.

The MATLAB Computer Vision and Robotics System Toolboxes provide the tools needed for stereo image processing, point cloud registration, as well as ROS integration. Additionally, the standard packages and visualization tools in MATLAB allow for rapid testing of various sensor fusion techniques, and algorithm iterations.

4.3.2 ROS Integration

The deployed system uses the ROS message framework for sharing data between nodes. Therefore, a significant objective of the reconstruction implementation is support for ROS integration. MATLAB's Robotics System Toolbox contains built in support of the standard ROS messages needed to integrate with the existing system. However, a drawback to MATLAB's ROS interface is a lack of built in support for message synchronization. For the 3D reconstruction algorithm presented in this thesis, it is assumed that the left image, right image, and pose at time step t , are at least approximately the same. Without synchronization

between left and right images, the disparity will be incorrect as the images are essentially misaligned. Without synchronization to the Pose messages, there will be errors in the location of re-projected world points. The standard ROS `Message_Filter` library addresses this issue with both an approximate time, as well as an exact time synchronizer. The message filter keeps a buffer of messages and returns the synchronized messages when the time stamps between messages are matched. Because MATLAB lacks this functionality, a simple approach was used to unblock progress towards system development. Algorithm 1 demonstrates how the issue of message synchronization was temporarily addressed in MATLAB.

Algorithm 1 ROS Message Synchronization in MATLAB

```

1: synced = false
2: poseBuffer  $\leftarrow$  pose_Buffer_Callback()           ▷ Pose published at higher rate
3: while not synced do
4:   left_Image  $\leftarrow$  latest_left_image_message       ▷ get latest image messages
5:   right_Image  $\leftarrow$  latest_right_image_message
6:   if new image not received then
7:     waitfor(rate)
8:     continue
9:    $\Delta t \leftarrow$  time difference between left and right image headers
10:  if  $\Delta t < thresh$  then
11:    pose  $\leftarrow$  get_pose_from_buf(stamp)           ▷ pose with closest matching stamp
12:    synced = true                                     ▷ time stamps close enough
13:    CurrentFrame.LeftImg  $\leftarrow$  left_Image
14:    CurrentFrame.RightImg  $\leftarrow$  right_Image
15:    CurrentFrame.Pose  $\leftarrow$  pose

```

The basic approach of Algorithm 1 maintains a buffer of pose messages using a standard callback framework. The pose messages are assumed to be published at a higher rate in this implementation. Therefore, multiple pose messages will be stored to compare to matched left and right image time stamps. Additionally, MATLAB's functionality to read the most recent message on a given subscriber is utilized. Once a subscriber has been created, at any point, the most recent message can be accessed. While the more common callback framework is also possible here, it was easier to develop a simple image synchronizer using this recent

message format. At each loop iteration, the time stamps between the left and right images are compared. If the time stamps match within a predefined threshold (i.e. $thresh = 0.1$ second), then the images are approximately synchronized. Once the image messages are synchronized, the pose buffer is checked for a nearest matching time stamp. The pose the with closest time stamp from the buffer is used and the current frame is updated with the synchronized messages.

4.4 Feature Matching

Speeded Up Robust Features (SURF) are extracted from the left camera grayscale image at each frame. The features between consecutive image frames are matched using the Sum of Squares Difference (SSD) method. After matching features, there are still some erroneous matches that need to be removed. This issue is particularly true for simulation environments where feature similarity is high due to duplication. Therefore, an additional filter is used to remove outliers based on odometry. The filter bounds are established by the change in motion between image frames as given by the FCU pose between frames. If the Euclidean distance between features is outside the filter bounds, then the matches are rejected.

Figure 4.3 demonstrates why feature match filtering is necessary even after the standard SSD feature matching algorithm is applied. Prior to filtering matches with IMU data, there are several erroneous matches between frames as in Figure 4.3a. However, after applying additional filtering techniques as discussed above, only good matches between frames remain as demonstrated in Figure 4.3b.

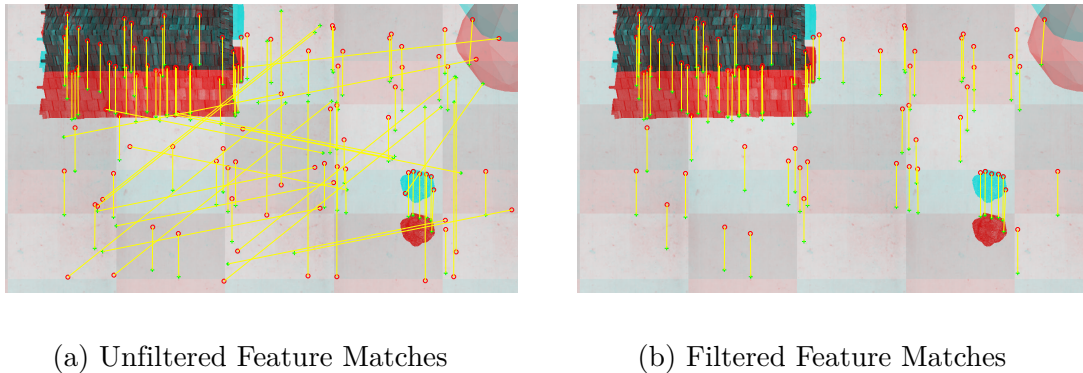


Figure 4.3: Previous frame features are depicted by + markers (cyan overlay). Current frame features are depicted by o-markers (red overlay)

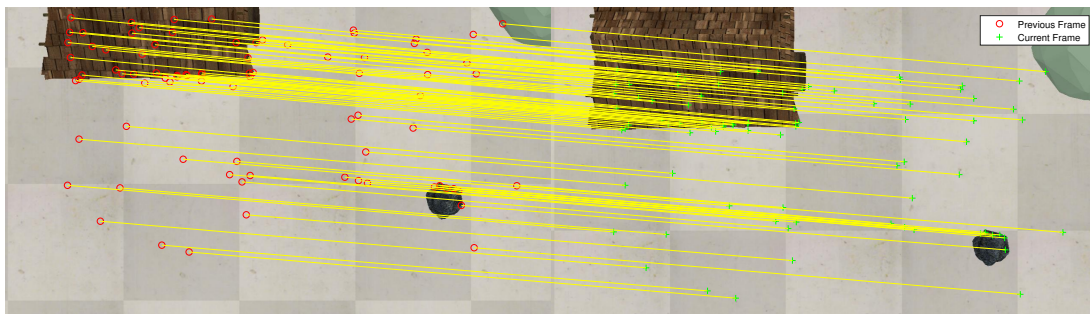


Figure 4.4: Side by side view of feature matching between frames.

4.5 Outlier Filtering

Outlier filtering is accomplished in two ways. First, outlier points are identified after the disparity is computed and 2D features are matched. At this point, before the 2D features are added to the sparse 3D cloud, the disparity at the associated pixel is compared to a minimum and maximum disparity threshold. If the disparity for the pixel associated with a matched 2D feature is outside these bounds, then that point is not added to the sparse feature clouds. Figure 4.5 demonstrates how noise in disparity images tends to present around object edges.

The vertical strip of dark blue (zero disparity) in the left fifth of the disparity image is simply due to a lack of correspondence with the right image (no image overlap in that region). The

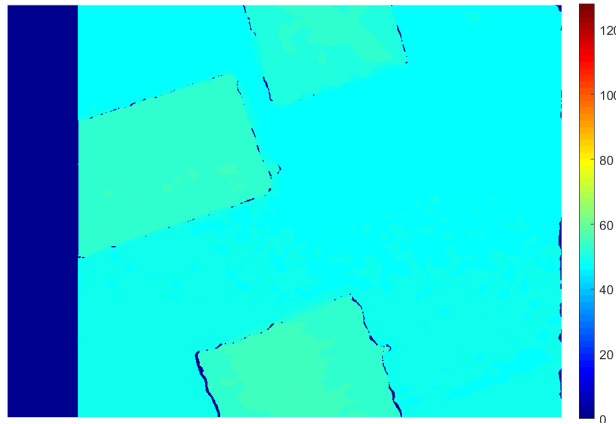


Figure 4.5: Disparity noise, due to specular reflectance and exposure, tends to occur around object edges

noise around the edges of objects, however, are likely due to specular reflectance and exposure issues. Therefore, what tends to present as good candidates for 2D features, are rejected due to disparity noise. To help ensure enough features are matched frame to frame, the feature extraction is performed using a uniform distribution approach to avoid clustering around edges.

Figure 4.6 illustrates uniform vs. non-uniform feature distributions. While both images contain the same number of extracted features (500), the typical non-uniform approach tends to collect around object edges where the gradients are largest. Conversely, a uniform distribution attempts to extract features across the entire image. Note that no features are selected in the left-most region of the images due to lack of overlap between the left and right camera images in that region.

The second form of outlier filtering compares the current pose estimate from the FCU to the estimate from ICP. If there is excessive error between the two, the frame is rejected, and a new set of data is obtained.

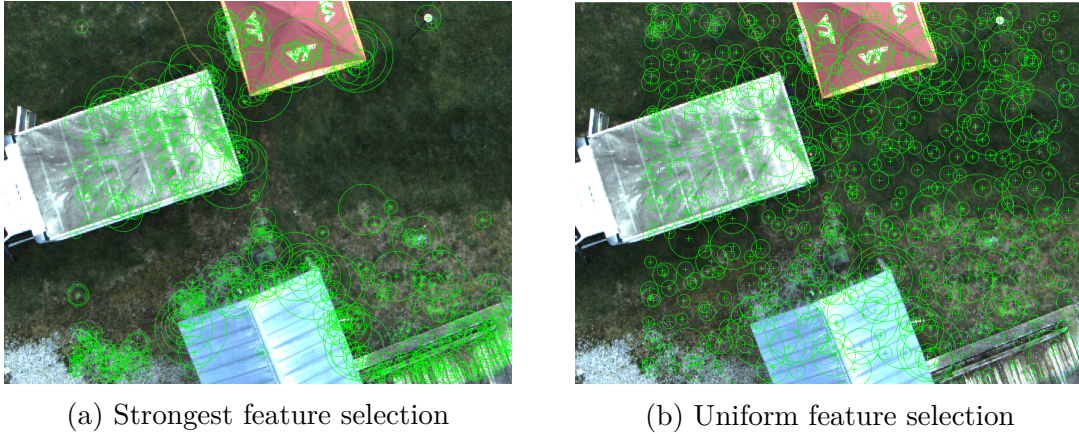


Figure 4.6: While both images have the same number of strongest features shown (500), (a) was produced using simply the strongest match algorithm. (b) was produced using a uniform distribution approach

4.6 ICP

ICP (Iterative Closest Point), is an iterative solver to approximate the rigid body transformation between two point clouds by reducing the root mean squared error between corresponding points. ICP is arguably the most common method for point cloud registration currently. In Section 2.2, the process of extracting and matching SURF features from 2D gray-scale images between the previous and current frame was described. Once good feature matches are determined, the 2D feature points are projected into 3D points using the the camera parameters described in Section 3.2. For the current frame and the previous frame, these 3D points represent sparse feature point clouds which are aligned using ICP.

For the purposes of discussing point cloud registration, the current frame may be referred to as the "moving" frame, and the previous frame as the "fixed" frame. This nomenclature is common for ICP point cloud registration where a rigid transformation is estimated to transform the moving point cloud to the fixed point cloud.

4.6.1 Initialization

While not strictly necessary, initializing the ICP algorithm with an initial guess can both improve accuracy, and reduce processing time. For this initialization, the change in UAV pose from the current frame to the previous frame is computed and passed to ICP as a predicted (initial guess) transformation to align the current frame point cloud to the previous frame point cloud. To construct this predicted transform, the change in both the translation and orientation estimates are determined directly from the flight controller pose estimates. The change between quaternion orientations, q' , from the current frame to the previous frame is found by

$$q' = q_2^* q_1 \quad (4.1)$$

where q_2^* is the quaternion conjugate of the current frame orientation, and q_1 is the unit quaternion of the previous frame orientation. As will be shown, for the affine 3D transformation matrix required to initialize ICP, the quaternion, q' , will need to be represented as a 3x3 rotation matrix. This conversion can be made for a given unit quaternion expressed in the form [5]

$$q = a + bi + cj + dk \quad (4.2)$$

Using the notation in Equation 4.2, the equivalent orthogonal rotation matrix, R , can be formed from the coefficients of the unit quaternion as shown in Equation 4.3 [5]

$$R = \begin{bmatrix} a^2 + b^2 + c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 + b^2 - c^2 + d^2 \end{bmatrix} \quad (4.3)$$

Additionally, determining the change in translation between frames is trivial and is given by

$$T = P_2 - P_1 \quad (4.4)$$

where, T is a 3x1 vector $[\Delta X, \Delta Y, \Delta Z]^T$ representing the change in position between the current frame, P_2 , and previous frame, P_1 .

Finally, MATLAB requires the predicted transform to be provided as an affine 3D geometric transform of the form

$$TF = \left[\begin{array}{c|c} R & 0 \\ \hline T^T & 1 \end{array} \right] \quad (4.5)$$

4.6.2 ICP Biasing

In practice, ICP can suffer from translation ambiguity due to noisy sensor. This ambiguity can also arise especially in low texture environments because feature points may have moved either due to small translation or due to small rotation. It is not always straight forward to determine what has caused the new pose (rotation, translation, or both) when the depth information is noisy. While initializing the ICP algorithm with an initial transform (as discussed in section 4.6.1) can help obtain a better estimate of the transformation between point clouds, this alone may not be sufficient to eliminate the ambiguity when the measurements contain significant noise. In this case, additional weighting can be applied to favor translation by adding bias points.

Bias points are simply a number of duplicate points placed in both the moving and fixed frames to add weight to translation during ICP estimation. There are several approaches to adding bias points. The approach found most effective in was to place bias points in

the fixed frame at the origin $(0, 0, 0)$, and points to the moving frame at the location of expected translation. The expected translation is defined the same as the difference between the previous estimated position, and the current position. In this way, the bias points are also correcting towards the GPS constrained UAV position. Figure 4.7 illustrates how bias points are added to the UAV camera pose locations in both the fixed and moving frames.

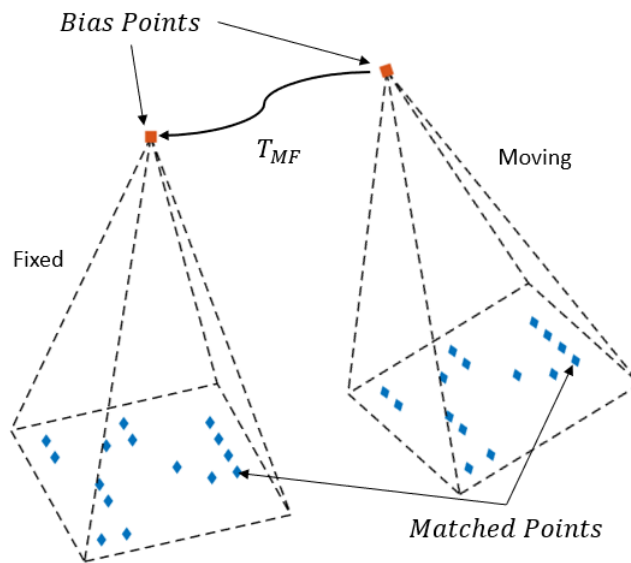


Figure 4.7: Bias points add weight (favor) the translational component of the ICP algorithm.

The number of bias points added to the sparse point cloud is determined by an integer translation weighting factor, k . The value of k directly represents the multiplicity of points to be added to the point clouds. This weighting factor can then be tuned to add more or less favor towards the expected translation by increasing or decreasing k respectively.

4.7 Kalman Filtering

While adding bias points to the moving and fixed point clouds can improve the translation estimate from ICP, the method also introduces an undesirable roll bias that accumulates into

a "twisting" of the reconstructed scene. This roll bias can be corrected for using a Kalman filter.

The Kalman filter, invented by Swerling (1958) and Kalman (1960), is a linear Gaussian filtering technique [24]. The Kalman filter is a probabilistic or belief-based approach using the concepts of prediction and correction. In general, the Kalman filter algorithm can be presented as a set of prediction (a priori) and correction (posteriori) equations as in Algorithm 2 [21].

Algorithm 2 Kalman Filter

- 1: $\hat{x}_k^{(-)} = A\hat{x}_{k-1} + Bu_{k-1}$
 - 2: $P_k^{(-)} = AP_{k-1}A^T + FQF^T$
 - 3: $K_k = P_k^{(-)}C^T [CP_k^{(-)}C^T + R]^{-1}$
 - 4: $\hat{x}_k = [I - K_kC]\hat{x}_k^{(-)} + [K_k]y_k - [K_kD]u_k$
 - 5: $P_k = [I - K_kC]P_k^{(-)}$
-

The superscript, $(-)$, and subscript, k , in Algorithm 2, denote the a priori and the discrete time index respectively. Therefore, lines 1 and 2 in Algorithm 2 describe the prediction (a priori) step of the filter where \hat{x} is the 6-DOF $(X, Y, Z, \alpha, \beta, \gamma)$ state estimate, and P is the covariance matrix. In the state estimate, position is expressed in terms of x-, y- and z-coordinates, and orientation is expressed as the Euler angles roll, pitch, and yaw (α, β, γ) .

In this thesis, the predicted state, is the state estimate obtained by using ICP, and the correction is the GPS constrained UAV position estimated from the flight controller. In this way, the Kalman filter equations are simplified by the fact that the motion and observation models become identity. Further simplifying the solution, is the lack of control input, u . Additionally, the process and measurement noise covariance matrices are assumed constant. The measurement and process noise are chosen as tunable parameters for the Kalman filter and for a given set of covariance matrices, the Kalman gain, K , will converge to a constant.

Therefore, the Kalman filter algorithm as implemented can be simplified to Algorithm 3.

Algorithm 3 Simplified KF Implementation

- 1: $\hat{x}_k^{(-)} = \hat{x}_{icp}$
 - 2: $P_k^{(-)} = P_{k-1} + Q^*$
 - 3: $K_k = P_k^{(-)} \left[P_k^{(-)} + R \right]^{-1}$
 - 4: $\hat{x}_k = [I - K_k] \hat{x}_k^{(-)} + [K_k] y_k$
 - 5: $P_k = [I - K_k C] P_k^{(-)}$
-

4.8 Scene Reconstruction

Finally, after applying the steps outlined in the preceding sections, the scene is reconstructed by transforming the current frame point cloud from the camera frame to the global frame using the best estimated pose from Kalman filtering. The points are then kept from each frame to accumulate the 3D scene as data is processed. Since this is a frame by frame pipeline, there is no global correction step or optimization to alter the 3D points once they have been placed in the scene.

The same pose estimate used to project the 3D points into the global scene, may also be published to the UAV's on-board computer to use for projecting points into the cost map. In this way, the frame by frame pipeline is maintained, and the off-board system can be run online with the existing system to provide an improved estimate of the 3D scene.

Chapter 5

Results

In this chapter, the results of the reconstruction are presented. First, the results from data obtained from the simulation environment are presented followed by the results of processing on real data collected in the field.

5.1 Simulation

As discussed in Section 3.5.1, The simulation pattern is a 3-pass lawn mower style test run where the camera altitude is 22 meters. Images and ground truth positions were saved every meter of traversed distance of the camera. This section presents the results of the reconstruction performance on the simulated environment and demonstrate the evolution of filtering techniques (biasing, Kalman) employed in this thesis.

5.1.1 Evolution of State Estimation

The simulated data was processed using varying degrees of filtering techniques discussed in Chapter 4 to emphasize the affect each step has on reducing drift and position errors in the state estimate. Since the simulated data is noiseless, the ground truth position is in fact the true position. In other words, the reconstruction could be produced using only the ground truth position and the camera parameters for the theoretical/simulated camera. Therefore,

in this case, the estimated position should match as closely as possible to the ground truth position as a measure of “correctness”.

Figure 5.1 demonstrates the issue of drift which results from only relying on feature matching. Note that although there is little noise in the simulated data, there are still errors associated with the occasional erroneous feature matches, spectral noise from simulated surface effects, as well as ICP tolerance. This drift highlights the issue of translation vs. rotation ambiguity as well

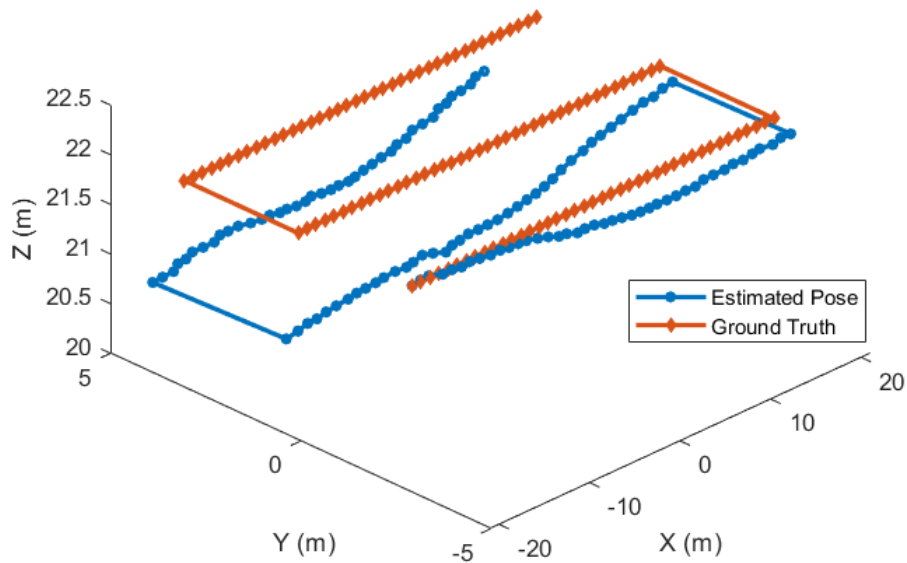


Figure 5.1: Estimated position using only feature matching

On the other hand, Figure 5.2 shows that with ICP biasing, the position estimate is improved over feature matching alone. Overall the errors are much smaller with biasing than without. However, despite these improvements, the biasing alone does not completely resolve the errors as the approach does not provide any correction to the data. Therefore, the estimate is still prone to accumulating errors.

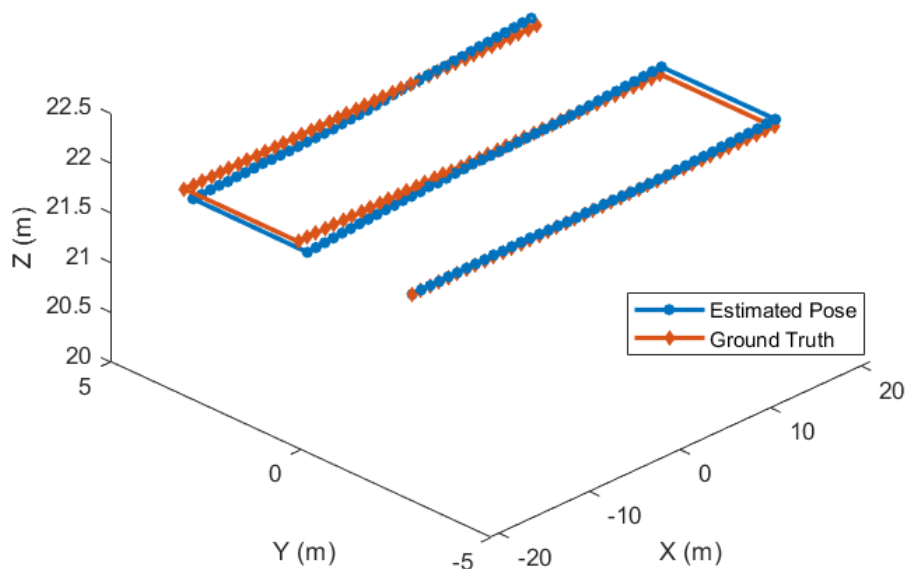


Figure 5.2: Estimate with ICP Biasing

Finally, the Kalman filter is added with feature matching and biasing to obtain the best result for position estimation. In this case, the state estimate tracks well with the true pose throughout the entire image sequence. Figure 5.3 shows the state estimate compared to ground truth using feature matching, ICP biasing, and Kalman filtering.

As shown in the sequence of figures (5.1-5.3), each step in the filtering process improves the state estimate. The final stage estimate shows minimal errors with averages on the order of 5-mm or less thus validating the approach overall.

5.1.2 Reconstruction of Simulated Environment

As demonstrated in the previous section, the best result utilizes feature matching, Bias points, and Kalman filtering to produce the best estimate of position. Therefore, the scene

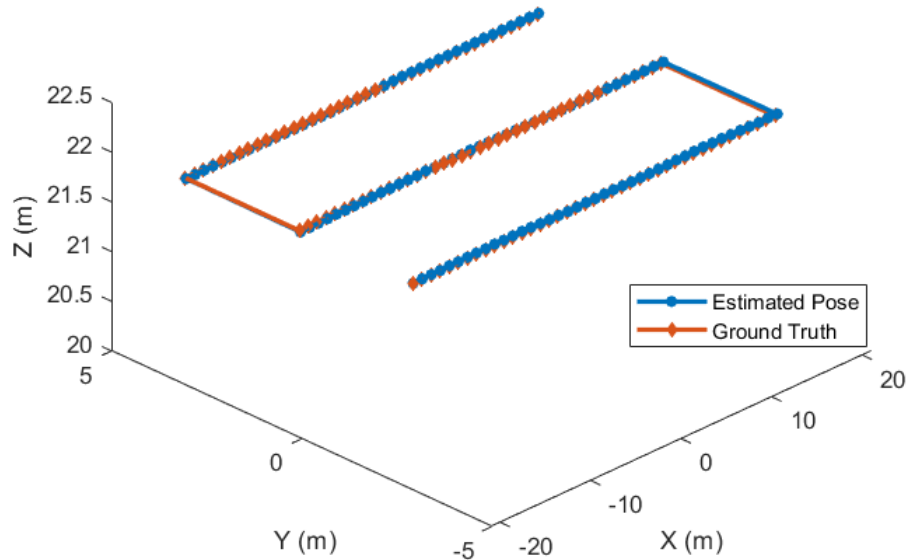


Figure 5.3: Estimate with Kalman Filtering

reconstruction shown in Figure 5.4 was produced using this combination of filtering techniques.

Overall, the reconstruction is of high quality with good local registration. The sparse regions around the edge of the reconstruction are due to de-noise filtering applied to the point cloud scene.

Reconstruction scale and alignment is verified by overlaying ground truth points with the point cloud scene. In Figure 5.5, the blue markers (+) represent grid corners on the simulation scene checkered floor while the red markers represent ground truth locations of the objects placed in the scene. As demonstrated by the resulting figure, the scale and alignment of the reconstruction is consistent over multiple camera passes without the need for frame to model registration in this case.

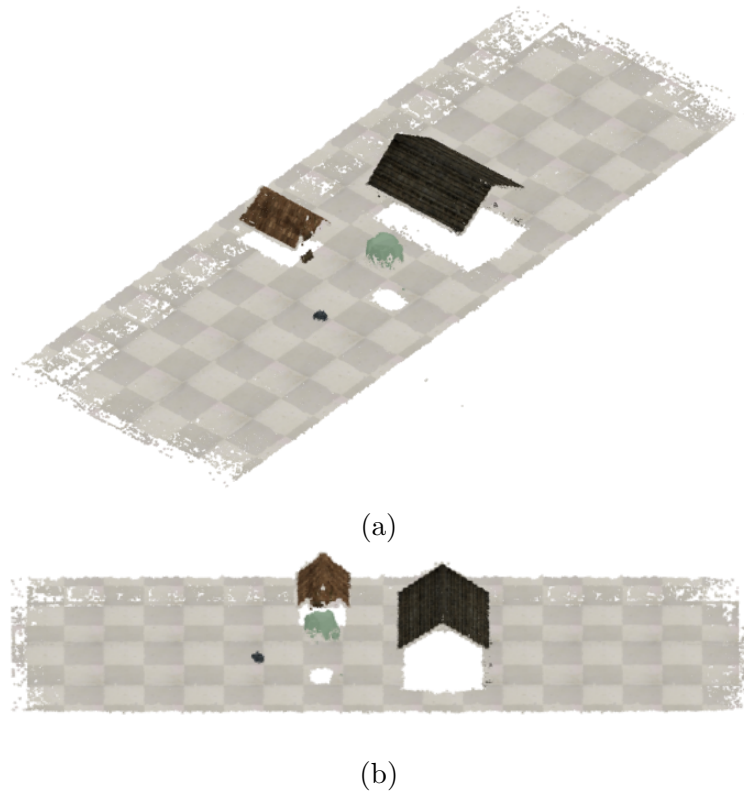


Figure 5.4: Reconstruction of simulated environment

To demonstrate depth accuracy from the stereo configuration, the ground plane (scene floor at 0-m reference) was extracted from the overall scene using point cloud plane fitting. Figure 5.6, shows the extracted ground plane which has an average z-height of about negative 5-cm. These differences between the expected and reported depth may be due to minor pixel correspondence errors which can occur when pixel similarity is high.

Despite the discrepancy in floor depth, the reconstructed scene is consistent overall with the simulation environment. This result indicates that the algorithm and filtering techniques are appropriate for the application proposed. However, as already discussed, the simulated data does not contain noise in ground truth data (fused with feature matched estimation) or sensor measurements (camera parameters completely known). Therefore, to further evaluate the reconstruction algorithm, real data is also analyzed.

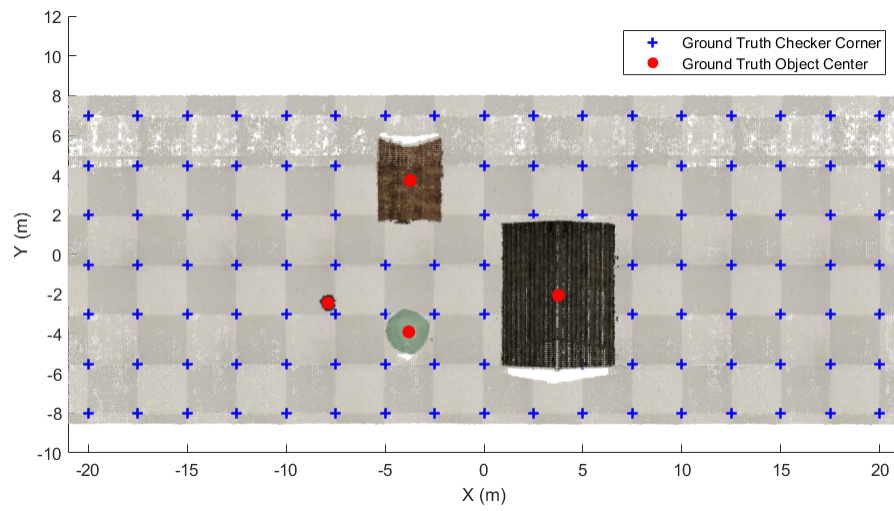
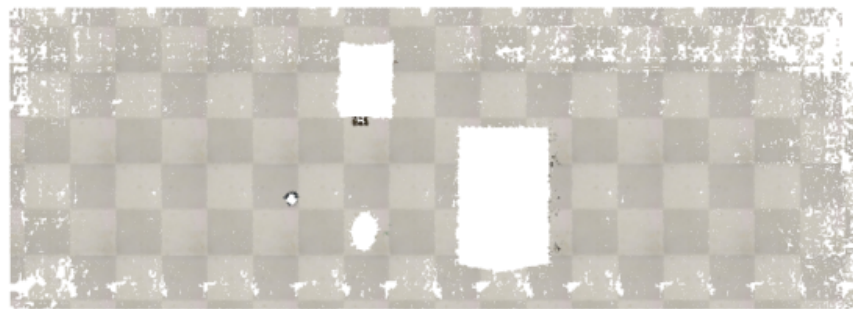


Figure 5.5: Reconstruction accuracy of simulated scene



(a)

(b)

Figure 5.6: Ground plane extraction from sim-reconstruction. (a) Top view, (b) front view of ground plane

5.2 Real World

The algorithm was also tested on real data collected from field tests as discussed in Section 3.5.3. Since the goal of this project is to improve on obstacle registration in the existing system, the results of the scene produced from the raw FCU position (existing system), is compared to the results using the algorithm presented in this thesis. Figure 5.7 shows a 2D orthomosaic (produced using Agisoft) of the images collected with specific objects labeled for reference. The visible scene spans about 143 meters in length, and 24 meters wide

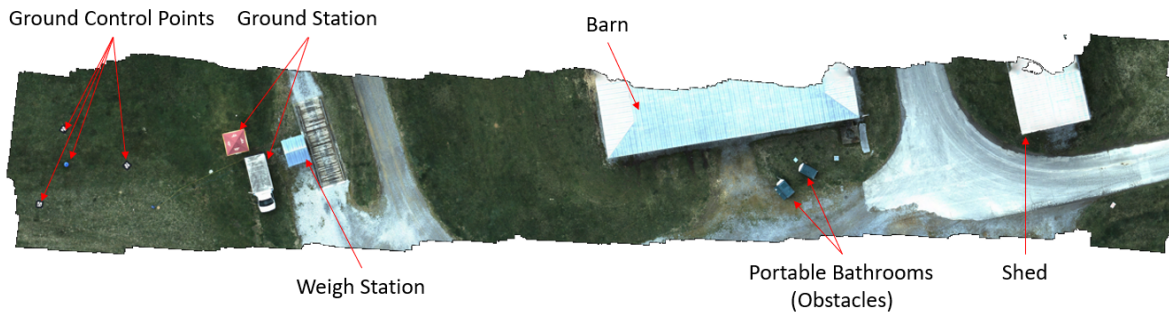


Figure 5.7: Agisoft 2D Orthomosaic of images taken during real data collection with labels identifying key objects in the scene

To compare the proposed algorithm to the raw results of the existing system, A single pass from the collected data is used to demonstrate frame to frame registration quality.

5.2.1 Scenes Compared

Overall, the reconstruction using the raw pose contains less clarity and significant obstacle dilation due to noise in the IMU sensor used for orientation estimation in the flight controller. Figure 5.8 and 5.9, shows the scene reconstruction from the collected data using only the FCU position compared to the reconstruction using the proposed algorithm. As demonstrated in Figure 5.8b and 5.9b, the level of detail is improved and there is more consistency in obstacle locations when feature matched estimation is fused with the raw pose.

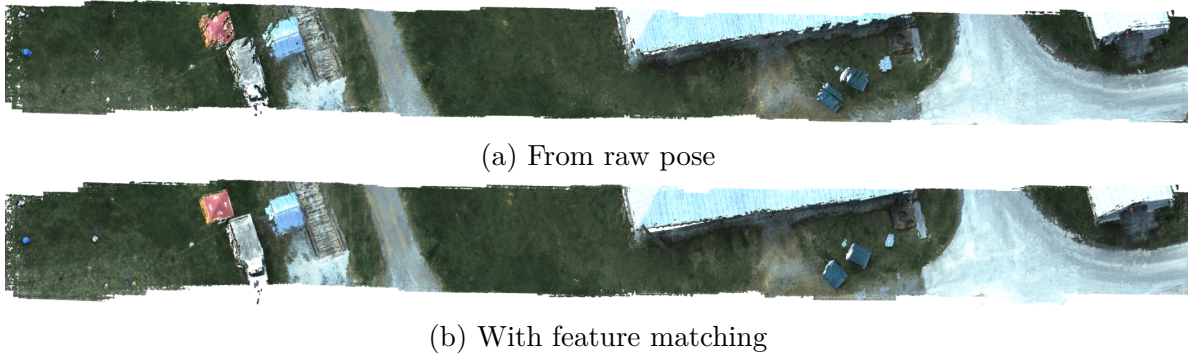


Figure 5.8: Top view reconstruction. raw FCU pose (a) compared to reconstruction with added feature matching (b)

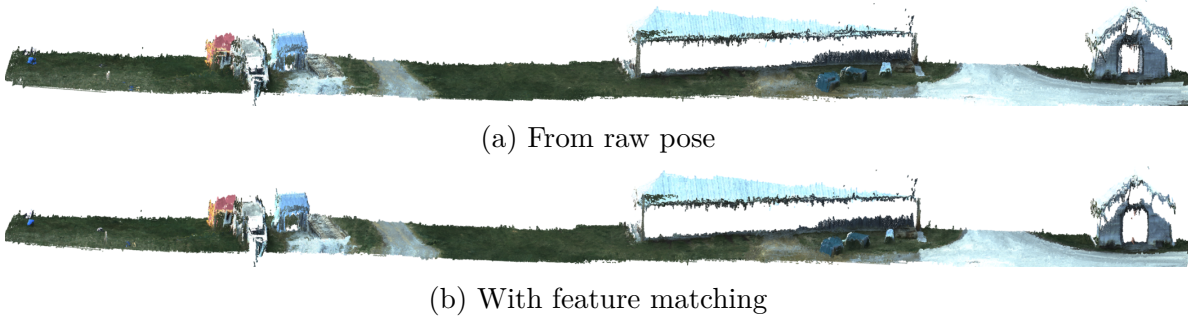


Figure 5.9: Front view reconstruction: raw FCU pose (a) compared to reconstruction with added feature matching (b)

To illustrate in greater detail the improvements made by fusing the pose estimate from the reconstruction algorithm with the flight controller, a segment from the overall scene is compared in Figure 5.10. Two segments are demonstrated in the figure: near the beginning of the third pass, and about 3/4 down. As can be seen from the segments taken without using the improved feature matched approach (images in the left column of Figure 5.10), the objects are duplicated or blurred in the scene. This occurs because the 3D points generated for each frame are projected into the scene using the orientation estimation from the IMU alone. This essentially has a similar affect to obstacle dilation in the scene where obstacles may appear larger than they actually are.

Conversely, object registration can be dramatically improved by fusing feature matched

ICP estimation with the FCU pose as proposed in this thesis. Figures 5.10b and 5.10d demonstrate how registration is improved using feature matched ICP estimation.

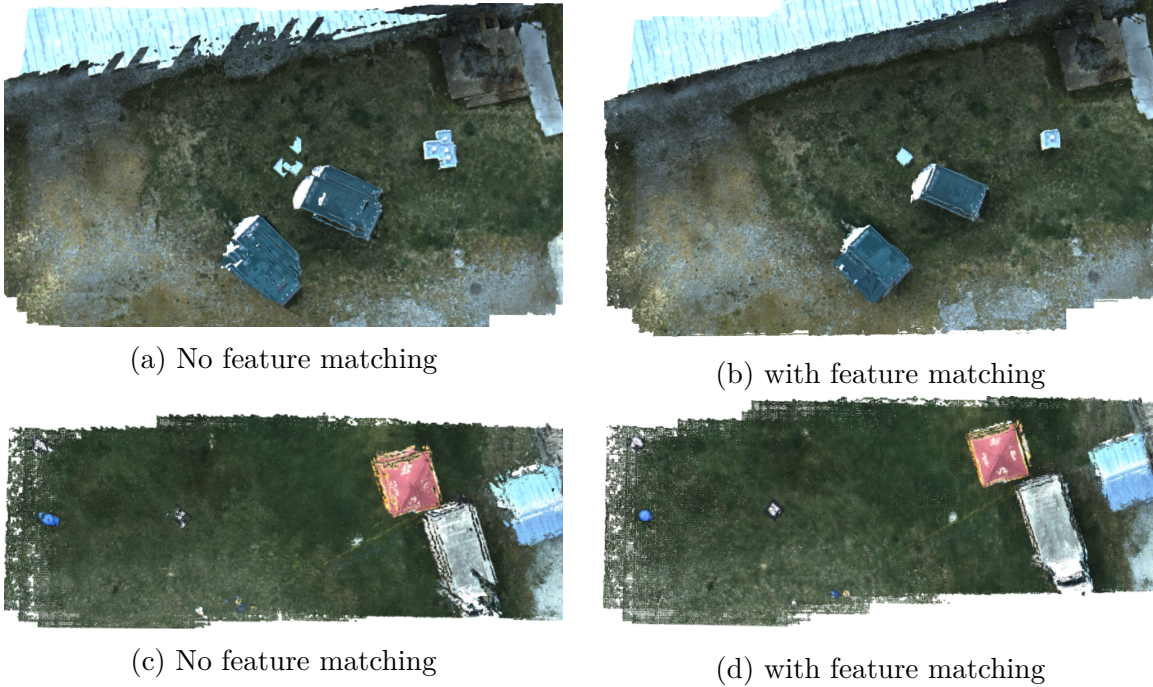


Figure 5.10: On the existing system using the raw position estimate of the FCU, object registration is poor (left column). However, when feature matched stereo estimation is fused with the FCU position, registration is improved in the scene (right column).

In these examples, it is clear that the method of feature matched ICP estimation can improve overall scene registration and quality. Additional reconstruction figures are provided in Appendix A.

Chapter 6

Discussion

In this chapter, errors associated with the reconstruction using the algorithm proposed in this thesis are presented. Additionally, the results are compared to a COTS product used to produce a reconstruction from the same data set. A brief discussion on future work is also presented at the end of the chapter.

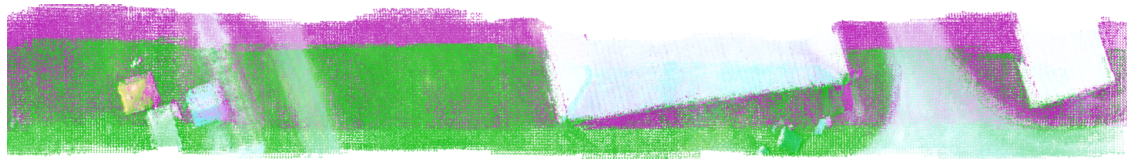
6.1 Errors

As presented in Chapter 5, certain aspects of the proposed reconstruction algorithm performed well compared to the current system. However, there are issues related to tracking for parallel passes which will be discussed in the following.

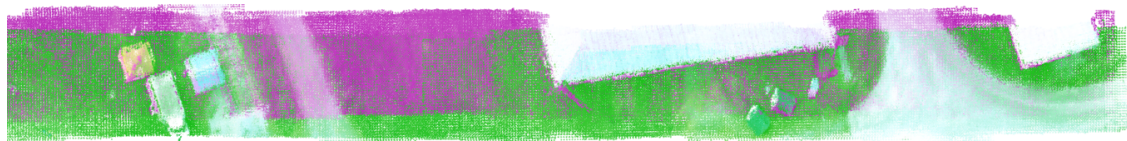
6.1.1 Tracking

When attempting to run through the complete data set, tracking issues arise in the transition regions between passes (either end of the scene) which results in a shift of the scene. Figure 6.1 demonstrates the tracking loss between the first pass and second pass of data. The point clouds from each pass have been shaded to illustrate the differences between the two. Most observable in the edges of the buildings, Figure 6.1 shows a shift in translation on each pass relative to the previous pass. Depending on filter settings (i.e. number of bias points,

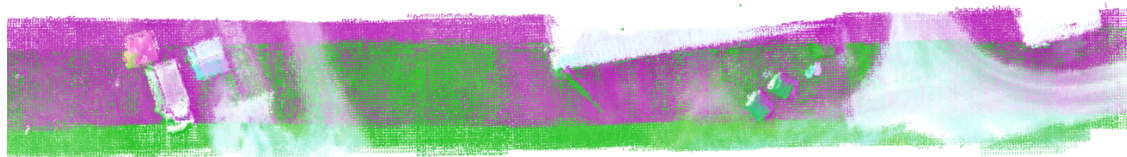
and Kalman Gain), these offset errors observed between passes have been measured in the Euclidean space at about 0.6 meters or less in magnitude.



(a) Pass 1 vs. Pass 2



(b) Pass 2 vs. Pass 3



(c) Pass 3 vs. Pass 4

Figure 6.1: When turning to begin the next pass (green), tracking is lost and results in a shift relative to the previous pass (magenta)

In general, this tracking issue seems to arise as a result of "jerk" when the UAV begins to transition from one pass to the next. The sudden movement of the UAV causes image blur and or large shifts in the observable field which can cause tracking to be lost completely. Additionally, poor feature matching due to either camera exposure or scene texture can also cause tracking errors to occur. Since there are no global landmarks used for registration in this implementation, recovering a loss of tracking to the model is not possible. When individual passes are observed alone as in Figure 6.2, relative scale is correct, and each pass contains reasonable detail and registration overall. However, when the complete scene is constructed as in Figure 6.3, there is degradation in scene registration overall.

Simply attempting to correct tracking errors by registering the frame to model does not work well. This is because of how the grid filtering applied to down-sample the point clouds

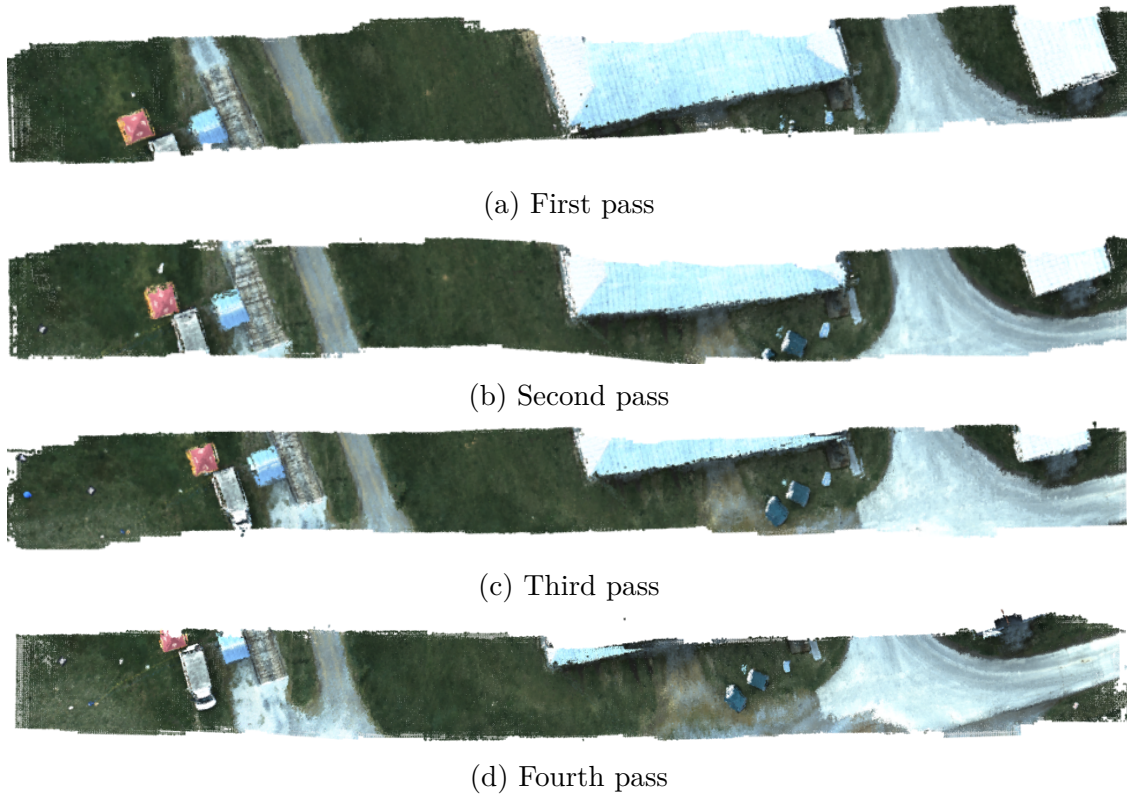


Figure 6.2: Data from each pass is used to reconstruct the scene segments

affects ICP. Without a precise initialization for ICP, it is easy to converge to an incorrect position by aligning points any number of grid steps away from the true position. However, it may be possible to add an additional correction step using RGB information to determine point correspondence in order to align the frame to the model. In this way, the bias due to grid spacing can be balanced by the inclusion of RGB information as well as XYZ location.

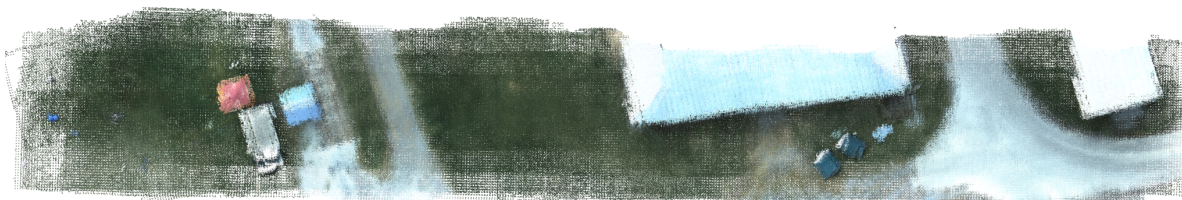


Figure 6.3: The complete scene reconstruction demonstrates tracking errors which lead to reduced registration quality

6.1.2 Position

Since the UAV position is constrained well with the RTK GPS, it is assumed that the UAV XYZ position approximates ground truth with reasonable accuracy (1-2 cm). Therefore, the estimated position from the proposed algorithm is compared to the UAV position to evaluate estimation performance in terms of XYZ position.

Figure 6.4 shows the error in position for the third pass. The error was calculated as the difference between the UAV position (ground truth) and the estimated position after feature matched ICP estimation is included. The data indicate that the errors are bounded with a max error of 0.625 meters.

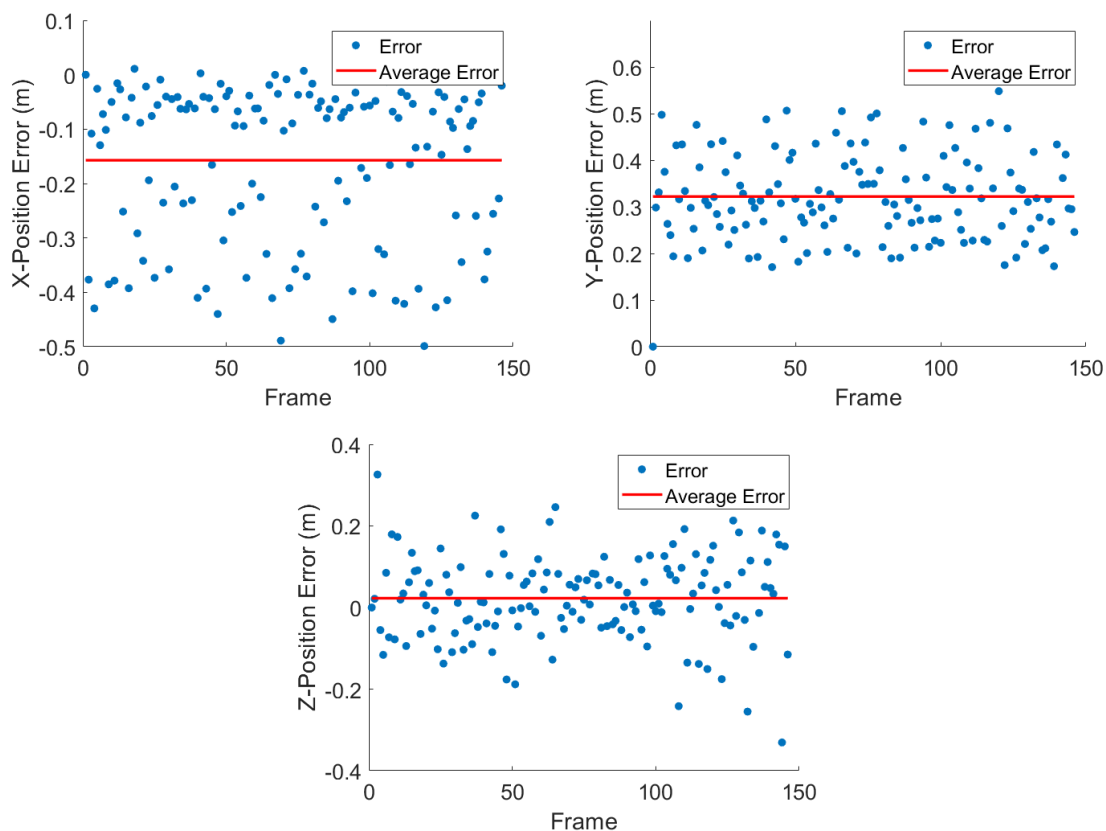


Figure 6.4: Position error between ground truth (UAV position) and the estimated position

The average errors shown in Figure 6.4, have about the same magnitude for all passes. Table

6.1 tabulates the average X-, Y-, and Z-error for all 4 passes. As can be seen by the low standard deviation, the magnitude of the error is similar across all the data.

Table 6.1: Magnitude of mean position error between ground truth and position estimate

Pass	Mag X-error (m)	Mag Y-error (m)	Mag Z-error (m)
1	0.159	0.323	0.025
2	0.164	0.330	0.002
3	0.157	0.322	0.023
4	0.154	0.317	0.004
Ave	0.159	0.323	0.013
St.Dev	0.004	0.006	0.012

It is interesting to note that there is an offset in the error (non-zero mean). This could be related to poor extrinsic calibration causing a consistent deviation from the FCU position estimate. Overall, these errors are within the same order of magnitude to what has been observed with the tracking errors as well. Therefore, some of the issues in multi-pass registration may also be related to position error overall rather than just tracking errors.

6.2 Agisoft

Agisoft Metashape is a COTS photogrammetry tool that can construct a 3D environment from 2D images much like the work in this thesis attempts to accomplish. Unlike the algorithm presented here, Agisoft is intended only as an off-line program that, in addition to feature matching for image alignment, maintains a pose graph of estimated camera poses. In this way, Agisoft employs pose optimization techniques which allow for global correction throughout the reconstructed scene. This step allows for corrections in projected points: a feature not developed in this work.

For the most part, the proposed algorithm maintained alignment throughout each pass of the scene relatively well with the exception of some small errors in the grassy region between the

barn and ground control station. This particular region also posed a challenge for Agisoft when attempting to estimate position in the same area. Figure 6.5 demonstrates which images Agisoft had trouble aligning.

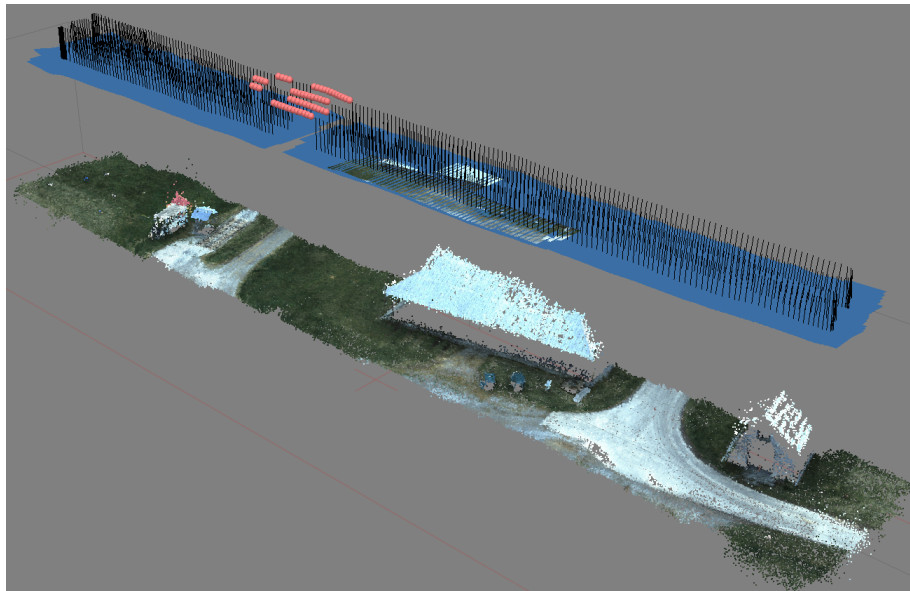


Figure 6.5: Agisoft tracking issues in grassy region between buildings: the pink dots indicate unaligned images

Despite these issues, Agisoft produced a well aligned, high quality 3D model of the scene as demonstrated in Figure 6.6.

6.2.1 Compared to proposed system

To compare the proposed system to the results obtained with Agisoft, ground control points (surveyed markers) were used to align the scene reconstructed from the proposed algorithm, to the same ground control points in the Agisoft reconstruction. Due to the tracking errors discussed in section 6.1.1, this was done for each of the passes independently rather than on the complete reconstruction as a whole.

Figure 6.7, shows each pass compared to the reconstruction produced with Agisoft. This

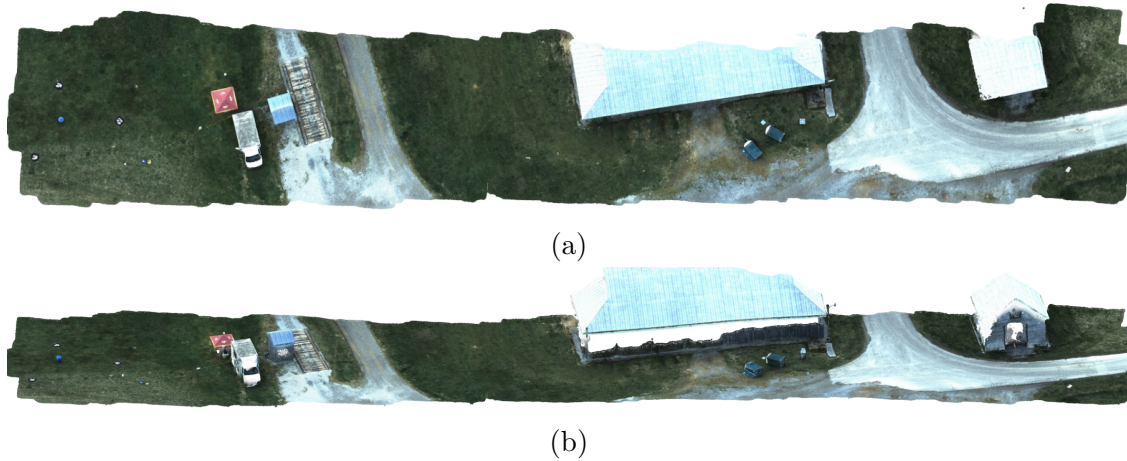


Figure 6.6: Reconstruction produced using Agisoft: Top view (a), front dimension (b)

comparison shows that overall, the match between the proposed algorithm and Agisoft is reasonable in terms of relative scale and position of objects in the scene. However, there are still observable errors in position and rotation for some of the objects. Specifically, in parts of the scene furthest from the control points (on the right half of the scene), position errors approach about 0.5 meters. Additionally, some of the objects are rotated by about 2 degrees on average. The rotation error is most discernible in the shed building on the far right of the scene.

While the proposed algorithm does not produce a perfect match to Agisoft, it is worth noting the comparison of computational efforts to produce each result. The average frame rate of the current implementation for the proposed algorithm is about 1-1.3 Hz, depending on specific filter settings. Conversely, Agisoft's average frame rate was about 0.03 Hz. That's a total of 6.3 hours Agisoft required to produce the results presented here.

Since the goal of this system is for online use, the results produced with the proposed reconstruction algorithm could be used even in a sub-optimal language such as MATLAB when low frame rate is not a concern. However, the same algorithm can be ported to another language for even faster processing time.

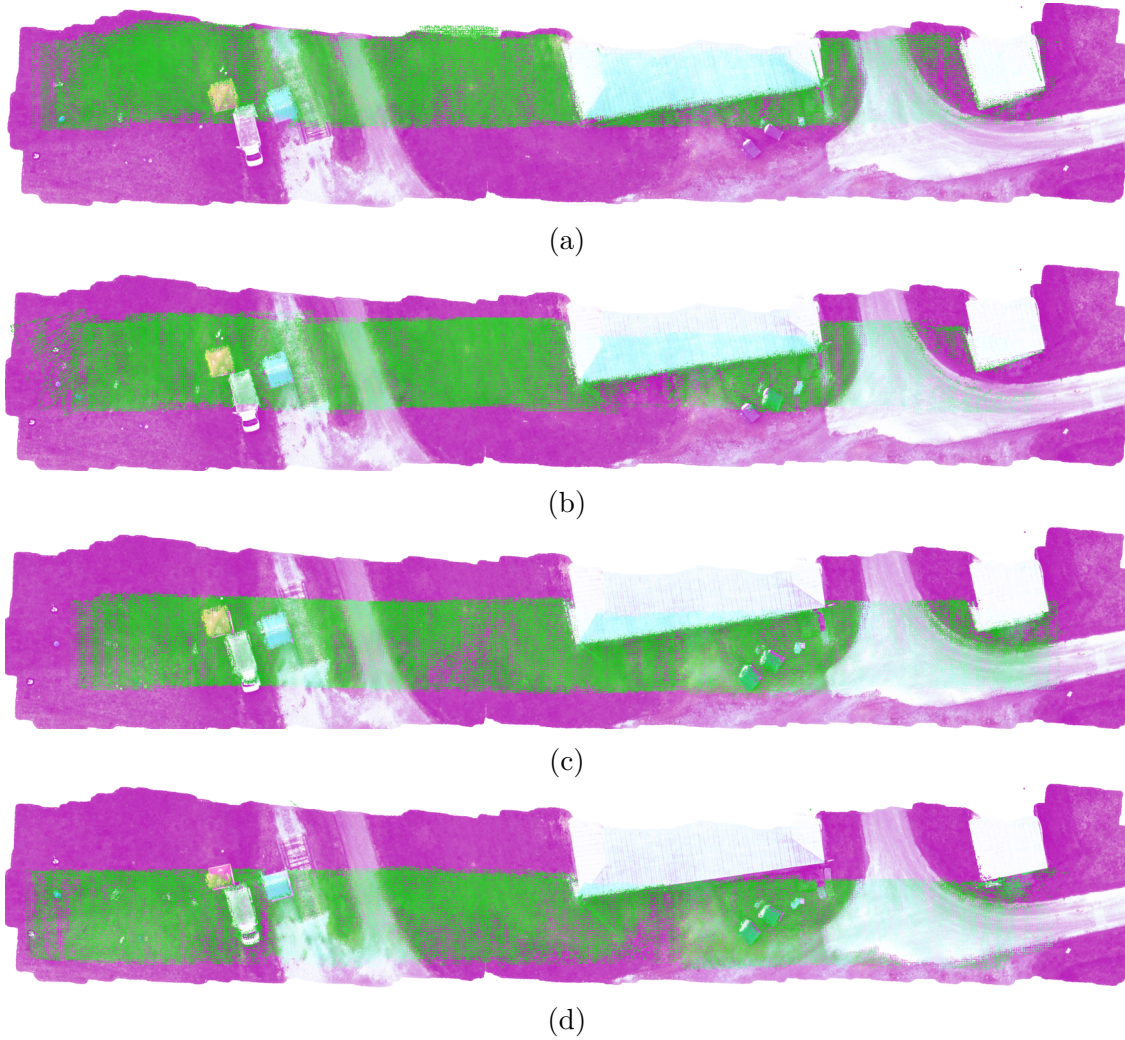


Figure 6.7: Reconstruction produced using Agisoft compared to the results produced from the proposed algorithm: first pass (a), second pass (b), third pass (c), fourth pass (d)

6.3 Future Work

For the reconstruction to be fully integrated and most effective on the existing system, the tracking issue needs to be resolved. It is clear from the results presented in this thesis that, while the addition of feature matched ICP estimation improves the frame to frame results compared to using the FCU pose estimate alone, there is still a need for frame to model matching and global pose correction. Frame to model matching could be made by including

RGB information to determine point correspondence for a frame to model ICP step following the initial frame to frame step. In general, including RGP information could improve the ICP method for frame to frame estimation as well while reducing the dependency on the accurate initialization transform for ICP. However, RGB information is hindered significantly as lighting changes occur from frame to frame.

Therefore, by leaning more heavily on the approach taken by many SLAM algorithms which perform a more robust key-point or landmark detection, the reconstruction results will be much improved overall. More importantly, SLAM also generally uses an optimized backend to correct a camera pose graph. This optimization backend is normally a bundle adjustment method to reduce reprojection errors between a set of poses with corresponding image points [27]. Such integration of an optimized backend would allow for the corrections to the reconstructed scene as multiple frames are captured. However, this approach would require additional work to reconfigure how the existing system utilizes the 3D data. Currently, the 2D cost map used for navigation only uses frame-by-frame point cloud data. Therefore, it would need to be modified to use bundled scene information to adjust. A logical next step to evaluate this theory, would be to test the sample data set with ORB-SLAM to compare the effectiveness of such an approach compared with the approach presented in this work.

Additionally, the global data from GPS may be incorporated into the optimization backend as done by Wheeler et al. [25]. In this work, the GPS information is used to correct global position when available, but allows a local map to be created in the front-end. This could improve the reconstruction results by allowing the front end to produce a strong locally aligned reconstruction, then, by including GPS information in the optimized backend, align the reconstruction in a global sense.

Finally, the MATLAB platform is great for development, but to improve speed for online integration, the program should be ported to C++ for increased performance. This can be

done using MATLAB's C-Coder; however, this step requires additional effort to make the MATLAB code more portable.

Chapter 7

Summary and Conclusions

In summary, this thesis describes a framework for fusing stereo odometric data with UAV pose data to obtain a 3D reconstruction. The proposed algorithm is intended to improve registration in the 3D scene for obstacle detection. The method relies on frame to frame matching of SURF features in the left camera images. These matched 2D features are projected into 3D points using stereo vision techniques to produce a set of sparse feature clouds. ICP is then used to estimate the rigid transformation between these sparse feature clouds. By accumulating these frame to frame transformations, an estimate of the camera pose can be determined at each frame and fused with the FCU estimate via Kalman filtering. The net result of this work is a position estimate which can be used to align a 3D point cloud into the global scene.

When compared to Agisoft, the reconstruction was reasonable in terms of relative scale and overall position when aligned using ground control points. The errors between Agisoft and the proposed algorithm were on the order of 0.5 meters or less in X-Y translation and about 2 degree of rotational misalignment on average. However, the processing time for the proposed algorithm is about 1-1.3 Hz on average which is fast enough for some online applications. Conversely, Agisoft is strictly an offline program with an average frame rate of about 0.03 Hz for the data set processed in this thesis.

Overall, the algorithm is shown to produce greater clarity and registration of the objects in the scene compared to using raw FCU pose alone. When comparing field data collected,

using the raw position estimate produces multiple instances of the same object in multiple locations due to noise in the IMU sensor. In the context of a 2D cost map, this means that obstacles appear much larger than they actually are. By adding the feature matched ICP estimation technique, frame to frame matching improves the registration of objects in the scene for sequential images. However, the algorithm suffers from tracking errors which can jeopardize scene quality for parallel passes where the same object may appear. This is largely the result of non-global pose correction/optimization as well as a lack of frame to model correction steps. Inclusion of RGB data for a frame to model correction step would improve these results.

while the results produced for single pass data is promising, frame to model matching need to be incorporated to fully realize the benefits of this method. Future work should focus on pose graph optimization and matching the frame to model using RGB information to reduce ICP ambiguity in point correspondence. Despite these challenges, this work has prepared a framework which can be improved and integrated into the system to improve global cost map generation.

Bibliography

- [1] Herbert Bay, Ess. Andreas, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF) Herbert. *European Conference on Computer Vision*, 3951:404–417, 2006. doi: https://doi.org/10.1007/11744023_32.
- [2] P Ben-Tzvi. Chapter 2. Kinematics. In *Advanced Robotics and Automation*, chapter 2, pages 1–25. Virginia Tech, 2018. doi: 10.1016/s0376-7361(09)70006-8.
- [3] Filippo Bergamasco, Andrea Torsello, Mauro Scavo, Francesco Barbariol, and Alvise Benetazzo. WASS: An open-source pipeline for 3D stereo reconstruction of ocean waves. *Computers and Geosciences*, 107(June):28–36, 2017. ISSN 00983004. doi: 10.1016/j.cageo.2017.07.001. URL <http://dx.doi.org/10.1016/j.cageo.2017.07.001>.
- [4] Mathias Bürki, Lukas Schaupp, Marcin Dymczyk, Renaud Dubé, Cesar Cadena, Roland Siegwart, and Juan I Nieto. {VIZARD:} Reliable Visual Localization for Autonomous Vehicles in Urban Outdoor Environments. *CoRR*, abs/1902.0, 2019. URL <http://arxiv.org/abs/1902.04343>.
- [5] R. P. Burn and R. P. Burn. Quaternions and rotations. *Groups*, pages 178–184, 2012. doi: 10.1017/cbo9781139163590.020.
- [6] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct SLAM for omnidirectional cameras. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:141–148, 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7353366.
- [7] Drone-Deploy. Drone & UAV Mapping Platform | DroneDeploy, 2019. URL <https://www.dronedeploy.com/>.

- [8] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3d reconstruction in real-time. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 963–968, 2011. doi: 10.1109/IVS.2011.5940405.
- [9] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.
- [10] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*, volume 2. Cambridge, 2nd edition, 2003. ISBN 9780521540513. doi: 10.1017/CBO9781107415324.004. URL <http://journals.cambridge.org/production/action/cjoGetFulltext?fulltextid=289189><http://scholar.google.com/scholar?q=related:zvPd pzI{ }WIIJ:scholar.google.com/{&hl=en{&num=20{&as{ }sdt=0,5{ }5Cnpapers2://publication/uuid/95851E6D-2EB4-48F0-B2F6-98D3C5BCD1D5>.
- [11] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. ISSN 09295593. doi: 10.1007/s10514-012-9321-0.
- [12] Ebrahim Karim, Siva Prasad, and Mohamed Shehata. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. *CoRR*, abs/1710.0:1–5, 2017. ISSN 1748-670X. doi: 10.1155/2017/7907163. URL <http://arxiv.org/abs/1710.02726>.
- [13] Prashant Kumar. *Online 3D Reconstruction and Ground Segmentation using Drone based Long Baseline Stereo Vision System*. PhD thesis, Virginia Tech, 2018.
- [14] Zhiliang Ma and Shilong Liu. A review of 3D reconstruction techniques in civil engineering and their applications, 2018. ISSN 14740346.

- [15] L. Maier-Hein, P. Mountney, A. Bartoli, H. Elhawary, D. Elson, A. Groch, A. Kolb, M. Rodrigues, J. Sorger, S. Speidel, and D. Stoyanov. Optical techniques for 3D surface reconstruction in computer-assisted laparoscopic surgery. *Medical Image Analysis*, 17(8):974–996, 2013. ISSN 13618415. doi: 10.1016/j.media.2013.04.003. URL <http://dx.doi.org/10.1016/j.media.2013.04.003>.
- [16] Annalisa Milella and Giulio Reina. 3D reconstruction and classification of natural environments by an autonomous vehicle using multi-baseline stereo. *Intelligent Service Robotics*, 7(2):79–92, 2014. ISSN 18612784. doi: 10.1007/s11370-014-0146-x.
- [17] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. ISSN 15523098. doi: 10.1109/TRO.2017.2705103.
- [18] Cosmin Popescu, Björn Täljsten, Thomas Blanksvärd, and Lennart Elfgren. 3D reconstruction of existing concrete bridges using optical methods. *Structure and Infrastructure Engineering*, 15(7):912–924, 2019. ISSN 17448980. doi: 10.1080/15732479.2019.1594315. URL <https://doi.org/10.1080/15732479.2019.1594315>.
- [19] Philip Ross. Lumotive Says It’s Got a Solid-State Lidar That Really Works, 2019.
- [20] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF Ethan. *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. ISSN 2380-7504. doi: 10.1109/ICCV.2011.6126544.
- [21] Steve Southward. Optimal Estimation and Kalman Filtering, 2019.
- [22] Shengjun Tang, Qing Zhu, Wu Chen, Walid Darwish, Bo Wu, Han Hu, and Min Chen. Enhanced RGB-D mapping method for detailed 3D indoor and outdoor modeling. *Sensors (Switzerland)*, 16(10):1–22, 2016. ISSN 14248220. doi: 10.3390/s16101589.

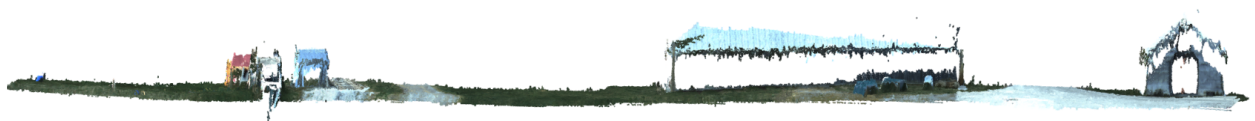
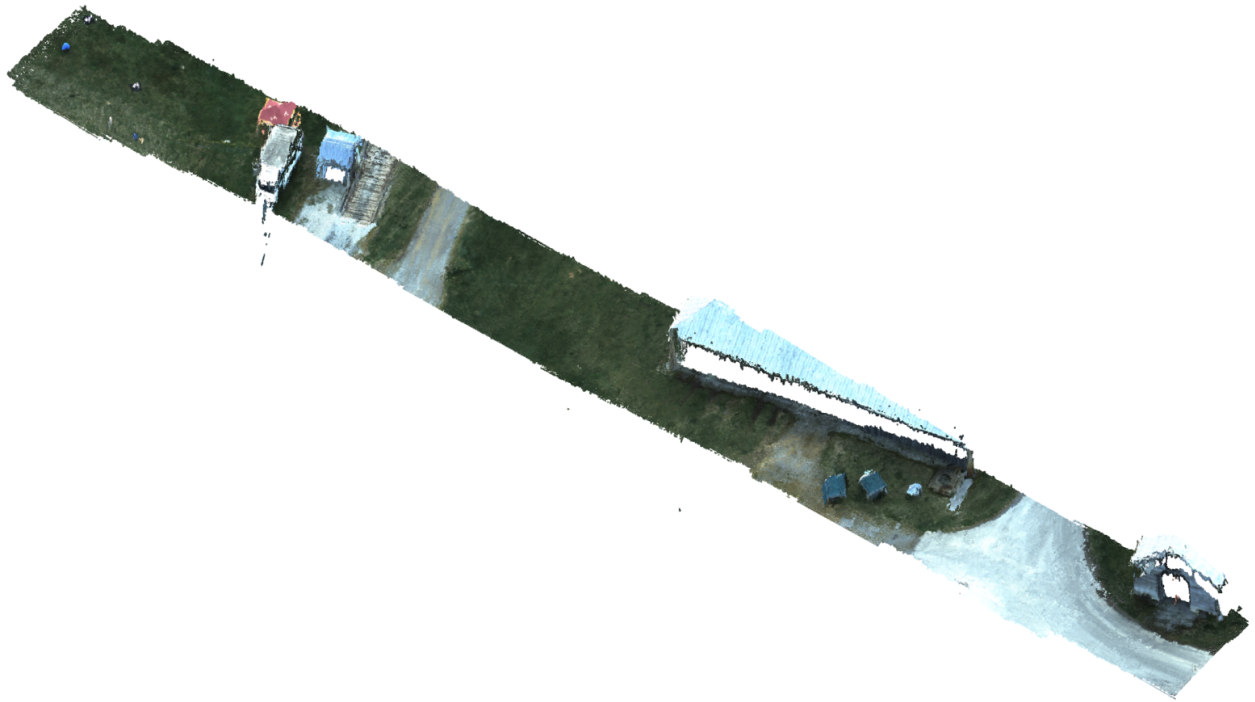
- [23] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Supplementary Material for: SEGCloud: Semantic Segmentation of 3D Point Clouds. *International Conference on 3D Vision (3DV)*, 2017.
- [24] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2006. ISBN 9780262201629.
- [25] David O Wheeler, Daniel P Koch, James S Jackson, Gary J Ellingson, Paul W Nyholm, Timothy W McLain, and Randal W Beard. Relative Navigation of Autonomous GPS-Degraded Micro Air Vehicles. 2017. URL <https://scholarsarchive.byu.edu/facpub/1962>.
- [26] Catherine Wong. How to Calibrate a Stereo Camera, 2018. URL http://wiki.ros.org/camera{_}calibration/Tutorials/StereoCalibration.
- [27] J Zhang, M Boutin, and D G Aliaga. Robust Bundle Adjustment for Structure from Motion. In *2006 International Conference on Image Processing*, pages 2185–2188, oct 2006. doi: 10.1109/ICIP.2006.312973.
- [28] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the Art on 3D Reconstruction with RGB-D Cameras. *Comput. Graph. Forum*, 37:625–652, 2018.

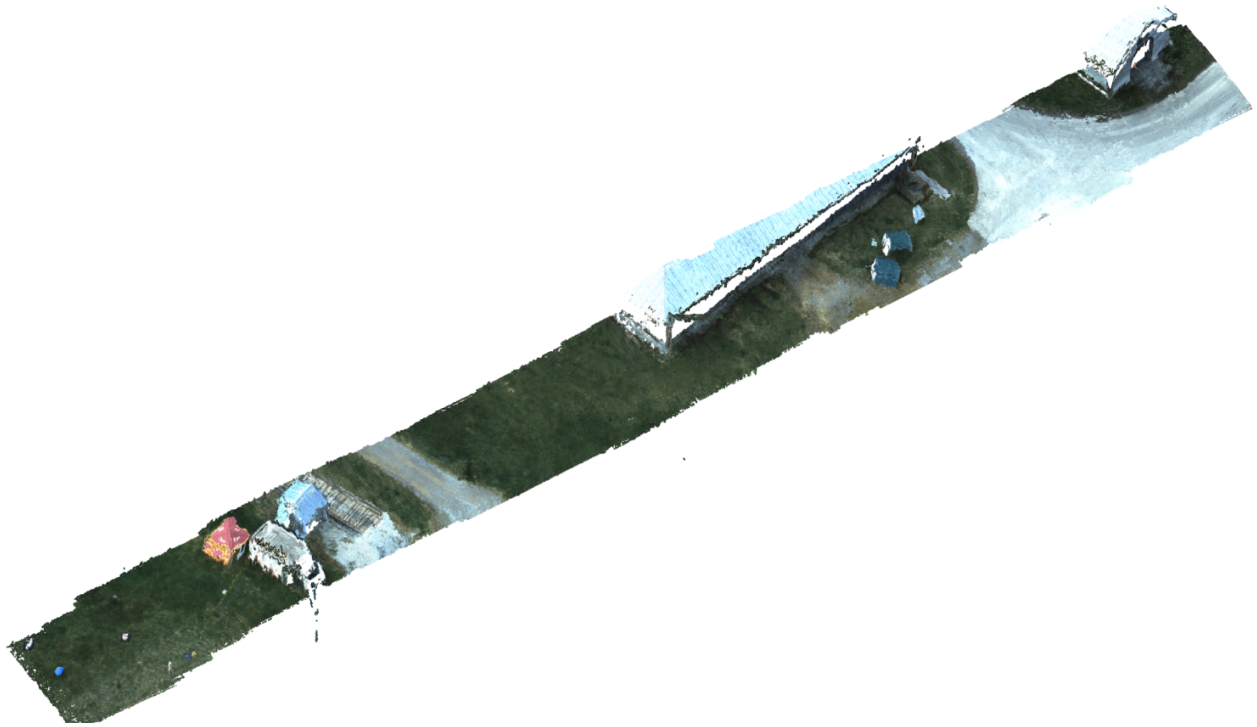
Appendices

Appendix A

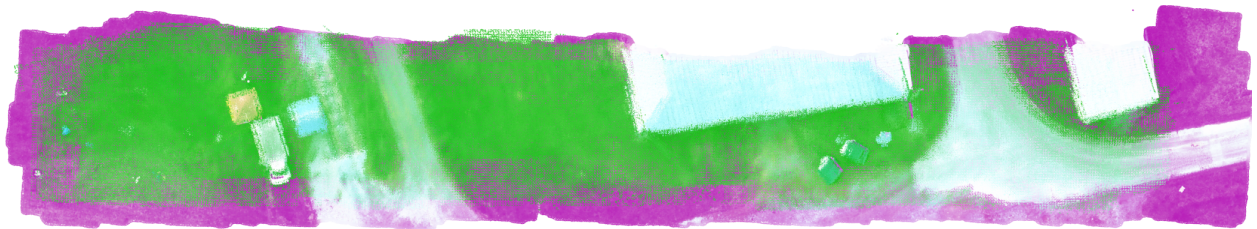
Additional Reconstruction Figures

The following three figures are additional views of the reconstruction from pass 3 with feature matching





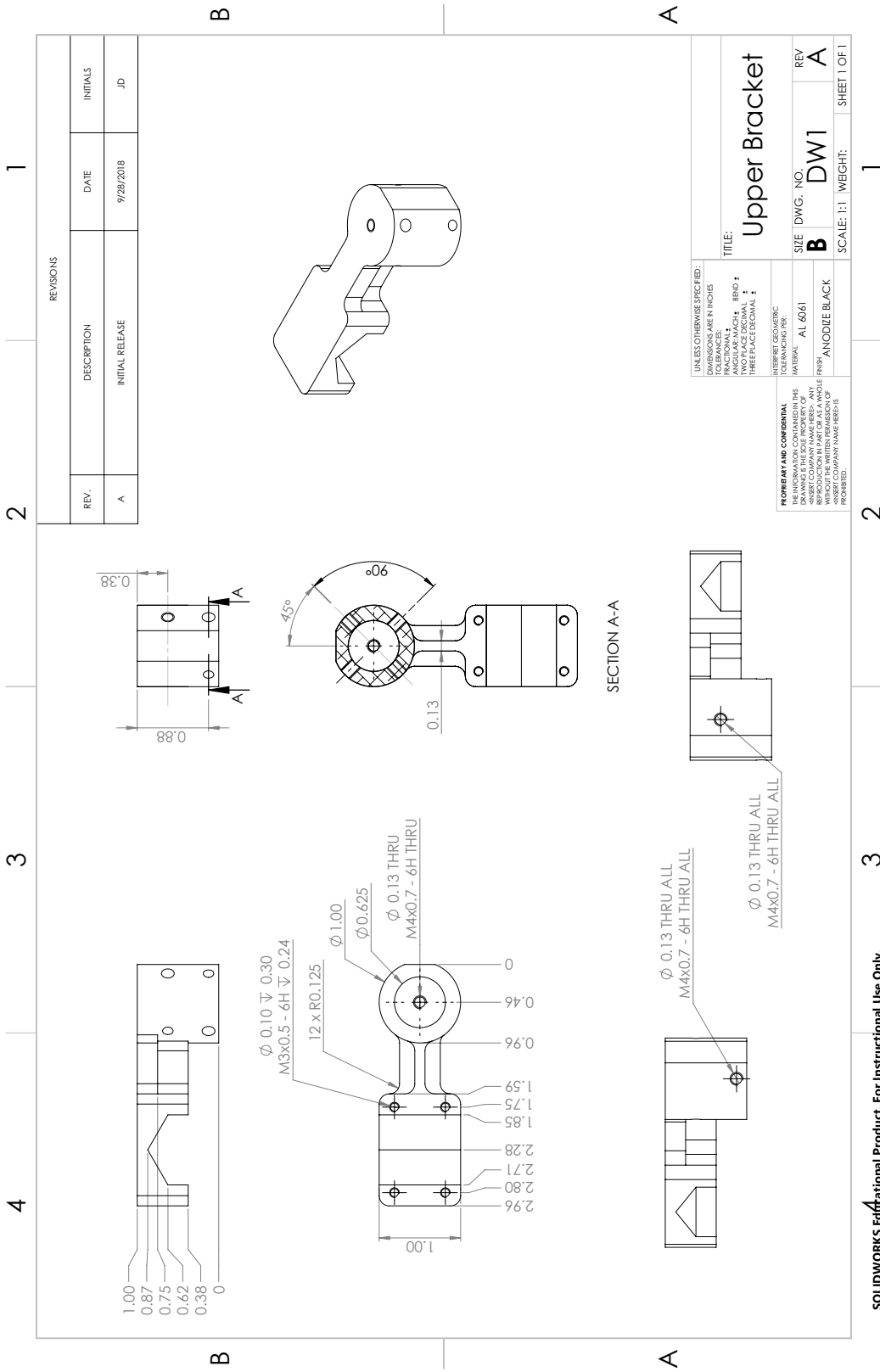
The figure below shows how the combined passes compare to Agisofts reconstruction



Appendix B

Mechanical Drawings

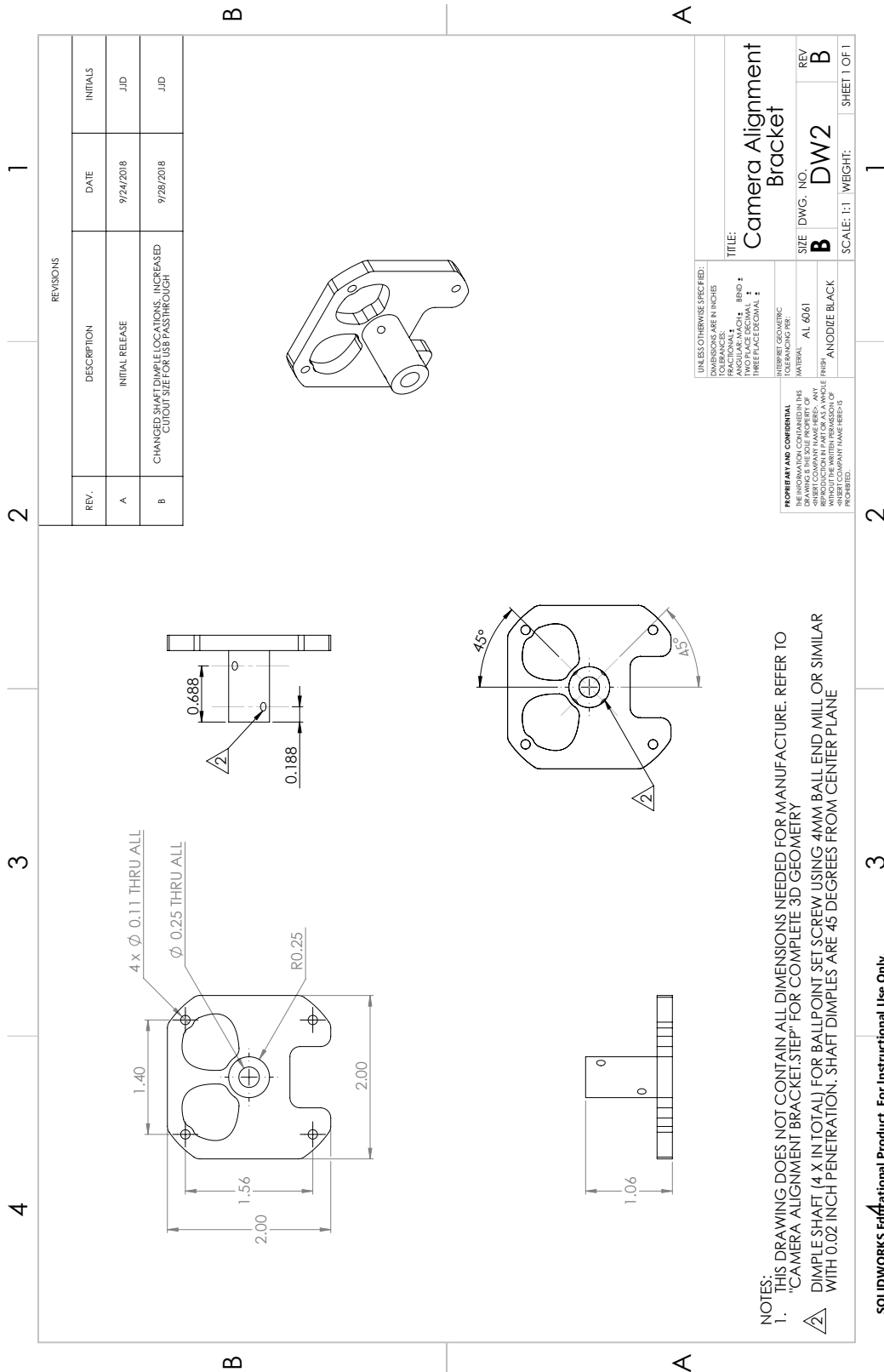
Mechanical drawings produced for the camera bracket components designed for the stereo camera boom presented in this thesis. Drawings are for reference and may not contain all dimensions needed for manufacture. .x_T files have been produced for manufacture purposes as needed.

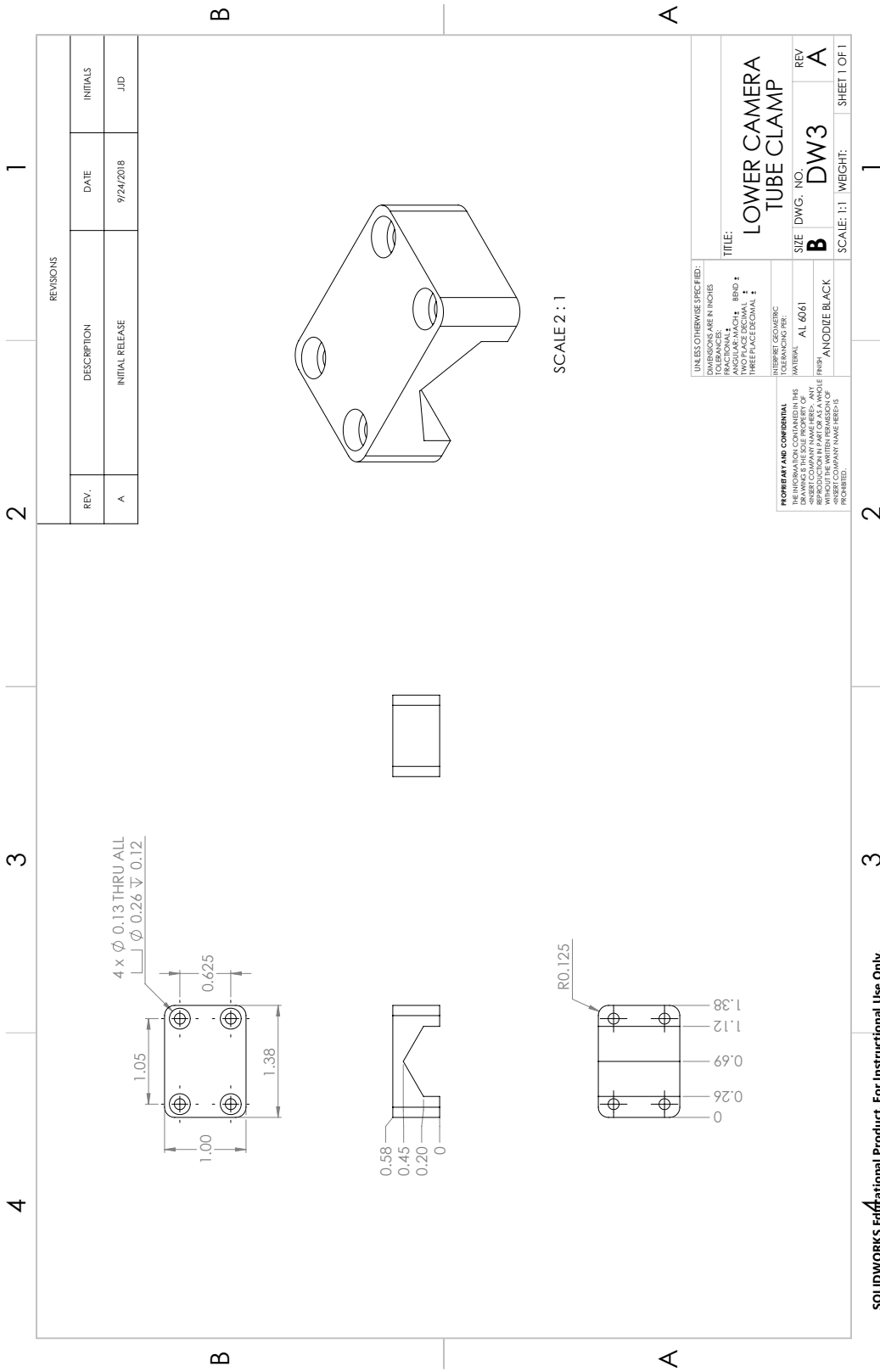


REVISIONS			
REV.	DESCRIPTION	DATE	INITIALS
A	INITIAL RELEASE	9/28/2018	JD

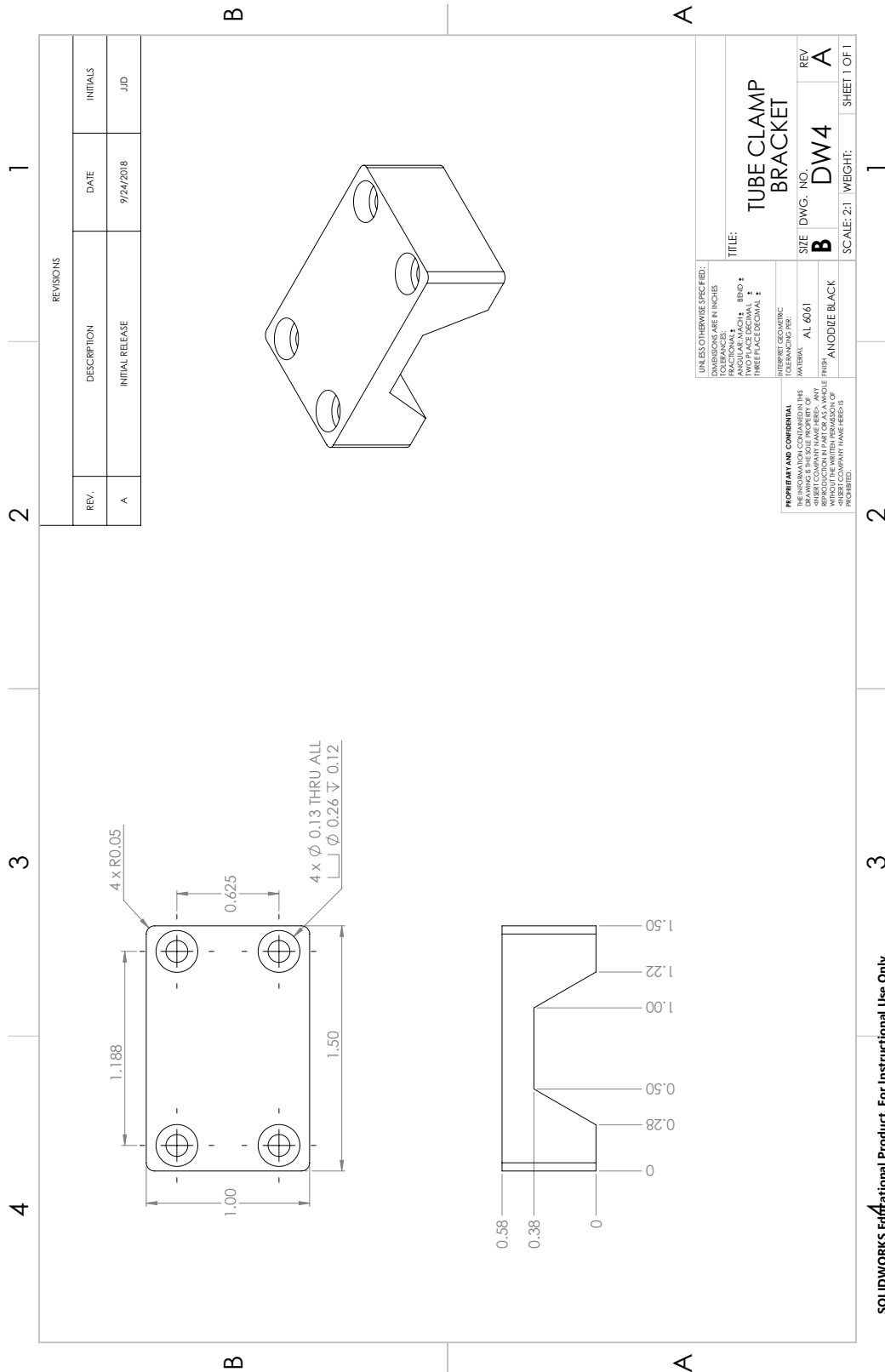
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES DIMENSIONS TO UNLESS OTHERWISE SPECIFIED: TOLERANCES: FRACTIONS: DECIMALS: ANGULAR: MACH. ± .0001 HOLE POSITION: ± .005 HOLE PLACEMENT: ± .005		TITLE: Upper Bracket	
PROPERTY AND CONFIDENTIAL INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF HEDERSON. REPRODUCTION OR TRANSMISSION OF THIS DRAWING WITHOUT THE WRITTEN PERMISSION OF HEDERSON IS STRICTLY PROHIBITED.		SIZE DWG. NO. B DW1	REV A
INTERPRET GEOMETRIC TOLERANCES PER ASME Y14.5-2009		MATERIAL: AL 6061	SCALE: 1:1
FINISH: ANODIZE BLACK		WEIGHT: 1	

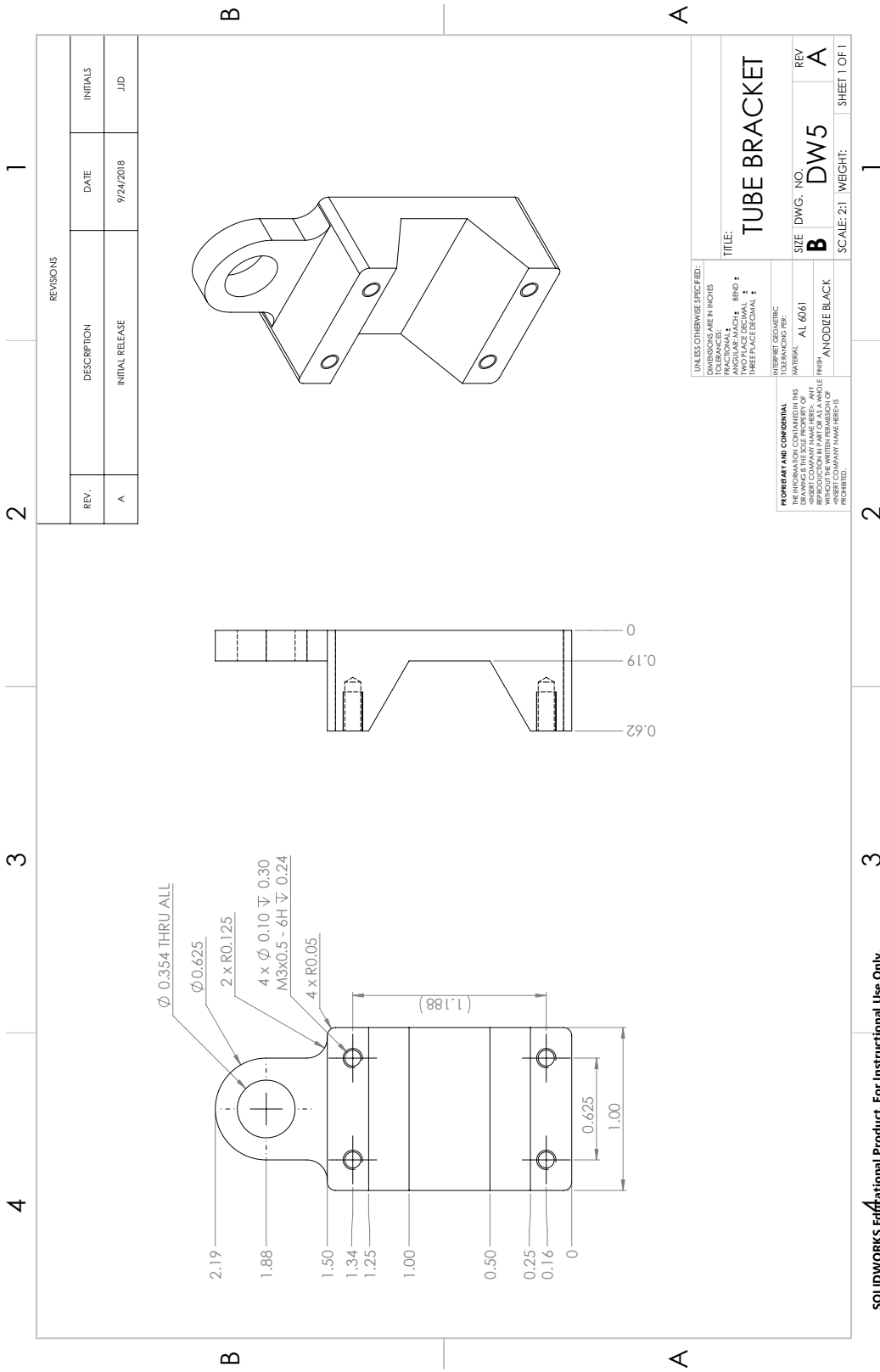
SOLIDWORKS Educational Product. For Instructional Use Only.





SOLIDWORKS Educational Product. For Instructional Use Only.





SOLIDWORKS Educational Product. For Instructional Use Only.