Fusion of Marker-based SLAM and IMU Sensor for Indoor Localization of a Quadrotor in a Mixed Reality Application

By

Naveen Gowrishankar

Thesis

Submitted in partial satisfaction of the requirements for the degree of

Master of Science

in

Mechanical and Aerospace Engineering

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Zhaodan Kong, Chair

Nelson Max

Xinfan Lin

Committee in Charge

2019

Thanks to UC Davis for this great learning experience.

Contents

List of Figures

List of Tables

Abstract

Fusion of Marker-based SLAM and IMU Sensor for Indoor Localization of a Quadrotor in a Mixed Reality Application

Augmented reality, Unmanned Aerial Vehicles and gaming are rapidly developing technologies in todays market. To support the development of a project that endeavored to combine these technologies, there was a need to build a framework for providing stable localization and control of a multiple drones in a large indoor structured GPS-denied environment and in the context of other hardware constraints. This thesis describes hardware and software architecture of the framework and also the implementation and testing of the localization system as part of the work done by the author for this project. First, Fiducial markers were used to implement a real-time vision based pose estimation system for the Solo drone using the Robot Operating System software architecture. The system capabilities were further expanded by implementing pose graph SLAM which allowed the drone to rotate and acquire a map of multiple markers while performing localization. Finally, this thesis investigates if the pose estimates from this marker based SLAM and data from an on-board IMU sensor can fused with a loosely coupled Extended Kalman filter in order to improve the scaling and accuracy of the pose estimate and surpass the limitations of the camera sensor. The results of the testing of the localization system at various stages are discussed.

Results show that this hardware limited, loosely coupled EKF implementation is able to generally track the pose estimate, perform in certain scenarios by rejecting high covariance estimates, offering the ability to run the filter at a high rate when compared to the SLAM system but ultimately offers no major performance gain in accurate positioning or scaling. A few methods and changes to the system that are likely to improve the tracking are discussed and suggested as future work.

Acknowledgments

I would like to thank all those people who helped me produce this work. First, I would like to thank Professor Nelson Max and Professor Zhaodan Kong for their continued guidance, patience and direction throughout this project. This project was instrumental in widening my horizons and picking up many new skills that I would not have been exposed to otherwise. I would also like to thank Weidong Guo, Nehal Agarwal, Gabriel Simmons and all the other members of the team with whom I had the pleasure of working on this multidisciplinary project. Finally, I would like to thank my friends and family for their encouragement and support along the way.

Chapter 1

Introduction

1.1 Motivation

Augmented Reality (AR) is a budding technology that is seeing increasing use and popularity in our day-to-day lives with the advent of mobile applications and games like Pokemon Go [1] and Snapchat[2]. Virtual reality (VR) has also come a long way, with many high-tech devices like the Oculus Rift[3], HTC Vive [4], PS-VR [5] etc, that have moved past the novelty stage and are now available in the market at competitive prices, allowing consumers to enjoy engaging virtual experiences and developers and artists to create software and entertainment experiences for these devices.

Mixed reality (MR) is an advanced form of AR where there is not only an overlay of information, but also direct or indirect user interaction with the virtual overlay. The Microsoft HoloLens [6], the most popular mixed reality focused device is currently relatively expensive, but it has opened up the market and now many companies have started to make their own AR and MR devices for specific applications. AR and MR, unlike VR, allow the users perspective to be firmly anchored in a real world coordinate frame while being able to access or control visual information from a overlaid virtual frame that can appear in fixed registration with the real world frame as the user moves or turns his/her head. AR requires use of sensors and software working together to create this immersive effect. AR can also be combined with many other existing fields and technologies where a mixed perspective reality is useful or can provide an engaging experience for the user such

as the medical industry where AR and MR tools are being developed [7] for use in surgery, patient information services and medical education. Other applications including, but not limited to, travel and sightseeing, advertisements and promotions, military vehicles and operations, maintenance and repair etc.

One of the future areas where AR can offer a paradigm shifting experience is with people who enjoy gaming and Unmanned Aerial vehicles (UAVs). There are precious few upcoming products that offer this mixture of technologies packaged into a indoor gaming product. Dr.Nelson Max, Distinguished Professor of Computer Science at University of California - Davis, proposed the idea of building a indoor multi-player experience where the users could see through a camera on their drones via a virtual reality headset and play games that moved the drone through a mixed reality world seen through the eyes of a flying machine under their control. To increase the immersiveness of the experience, feedback elements could be implemented, connecting the user, the machine and a mixed reality world together seamlessly. In this thesis, the first steps towards building a real world implementation of a system with this combination of technologies is described. This thesis also discusses results of experiments with various methods including off-board sensor fusion used to improve the basic functionality of mixed reality by providing real time localization of a drone in a mixed reality indoor structured environment.

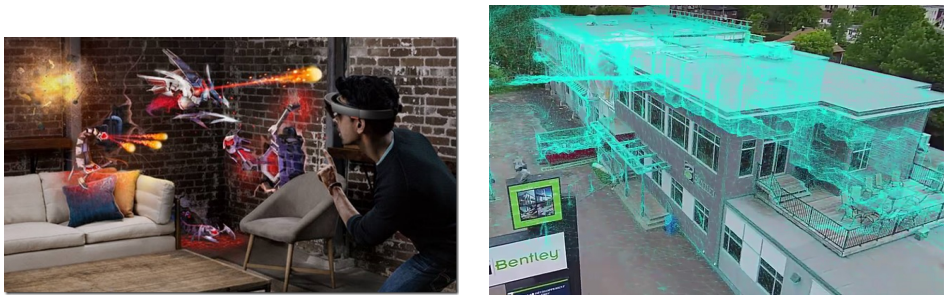


Table 1.1: Left: Depiction of Mixed reality using Microsoft HoloLens, Right: Illustration of Mixed reality from a drone point of view

1.2 Overview of technologies

1.2.1 Mixed Reality and gaming

The rise of Augmented reality started sometime around 2007, when it was recognized by MIT as one of the ten emerging technologies of 2007 [8]. Augmented reality technology, a super-set of mixed reality, has risen quickly [9] due to advancements in enabling technologies such as low cost sensors like the phone camera, progress in visual simultaneous localization and mapping, optics, and mature multimedia techniques, which allowed the growth of digital marketplaces for applications and games. Figure 1.1 conceptualized by Milgram and Kishino [10] illustrates how mixed reality and augmented reality go hand in hand with each other. In market terms, the usage of the terms varies slightly, with the term 'AR' typically describes a system that keeps the real world central but enhances it with other digital details, layering new strata of perception, and supplementing reality, while MR is used to describe a system where it is possible to interact with and manipulate both physical and virtual items and environments [11]. Traditionally, reality altering games have been perceived as a solitary technology, isolating a single individual in an artificial environment but research has shown that social interactions while gaming is considered a key factor for joyful gaming experiences [12]. The ideal for a good multiplayer mixed reality game is to create shared cognitive immersion and a sense of thrill and competition.

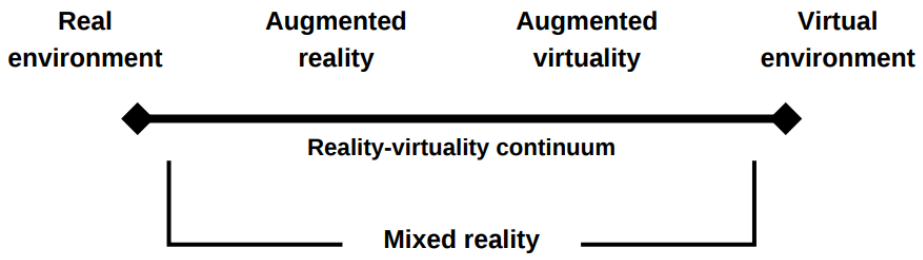


Figure 1.1: Real-Virtual Continuum [13]

1.2.2 Unmanned Aerial vehicles and Indoor localization

Unmanned Aerial Vehicles (UAVs) or drones, are a class of aircraft that do not have a human pilot on-board. They are becoming increasingly popular in various outdoor applications such as surveillance and reconnaissance. photography, mapping, and relaying of other forms of data collected from an altitude. Piloted quadrotor drones are also used for entertainment and sports purposes such as drone racing, which is drawing a growing audience. Although there are international competitions [14] for this kind of sport, drones for entertainment are currently very niche but hold a lot of potential when packaged with other ubiquitous and accessible media like computer games. The control scheme for piloting a quadrotor drone is quite similar to the controls that gamers are used to in first person view flight games. Indoor environments are quite challenging for drones due to need for additional sensors to replace GPS as the primary positioning sensor for outdoor UAVs.

Typically indoor localization systems for drones fall into one of the two broad categories based on where the sensors are located: On-board or Off-board sensors or a combination. On-board sensors are preferred when it is not feasible to mount sensors in the environment. A variety of On-board sensors have been used over the years to perform the important task of localizing a robot in a 3D environment.

Among On-board sensors, cameras are the most popular due to them being relatively lightweight, low power and data rich. The passive camera sensor tends to work well in feature rich environments, but does not perform well in featureless or visually degraded environments. Other sensors include active sensors such as 2D laser scanners (Hokuyo UTM-30LX) or rotating 3D scanners that can throw and detect infrared light bouncing off surfaces, RGB-D cameras such as the Microsoft Kinect, stereo cameras such as the ZED Stereo Camera, LIDAR and SONAR. Secondary sensors include typical drone mounted sensors such as the Inertial measurement Unit (IMU), Pressure sensor and magnetometer.

Ego-motion Estimation or the estimation of the motion or position of a camera system in an environment is an active research area that has numerous applications in fields such as autonomous vehicles and mixed reality systems. The camera sensor is usually

complemented with other sensors to improve the estimation process. This field is known as visual odometry. An overview of past work with these techniques is described in Chapter 5.

1.3 Project Overview and Goals

The goal of this project started with the idea of submitting a proposal to create an indoor exhibit for the SIGGRAPH conference 2018. SIGGRAPH is the annual conference on computer graphics convened by the ACM SIGGRAPH organization [15]. The exhibit would involve a large indoor space, approximately 30 feet long by 20 feet wide by 15 feet high with three quad-copters controlled by novice pilots in real-time using gaming joysticks. These pilots or players in this game would have only a first person view (FPV) of the arena through cameras mounted on their respective quad-copters. This first person view will be realized via a head mounted display that is also capable of rendering virtual elements onto the first person video feed, in order to immerse the players in this mixed reality arena. Different kinds of multi-player games can be potentially realized through this platform. Some early ideas included a 3D pong game with two players controlling two drones as paddles and another autonomous drone acting as the ball, a shooting game similar to Galaga, and a maze race game. The spectators outside the arena would be able to watch the quadcopters perform feats inside the arena and interact with each other. Another video stream could show a view of the game world as seen by the players including the virtual elements.

This kind of application for mixed reality and aerial vehicles is a novel idea that has not been fully implemented yet and has a lot of potential for spurring the interest in mixed reality gaming within a social, competitive and spectator sport framework. The goal to create an exhibit as described required efforts from both the Computer Science department and the Mechanical and Aerospace Engineering department. This thesis will describe my contribution as one of the core members involved in putting together the hardware and software architecture and investigating if using a loosely coupled EKF sensor fusion technique can improve the performance of a vision based Simultaneous Localization and

Mapping (SLAM) system for use in the mixed reality system.

1.4 Methods

First, the system architecture for this project is built using various hardware and software components. The description of the architecture and the rationale for various components is explained in chapter 2. A real-time, marker based vision based localization system using a Perspective and Point Algorithm (PnP) was implemented using the Robot Operating System framework and is discussed in chapter 3. A pose graph based SLAM algorithm was implemented to make the localization work in 6 degrees of Freedom in the arena, to allow for free movement of the quadcopter in the large space. The results of the SLAM algorithm are discussed in chapter 4. The limitations of the vision-only system drives the need to include the IMU sensor to better estimate the pose of the quad-copter. A sensor fusion algorithm based on the Extended Kalman Filter (EKF) is evaluated and the final results are discussed in chapter 5. Finally, the conclusion and future pathways towards improvement and other possible additions to this framework are presented.

Chapter 2

System Architecture

This section describes the various hardware and software components and how they are connected together to form a functional system.

2.1 Hardware

2.1.1 Flying Camera Drone - 3DR Solo

For this project, there was a need for a low cost drone with the capability to carry a camera that could stream High Definition video wirelessly to a PC with little latency and high reliability. The drone also needed to able to stay in the air while providing power to the camera for at least 15 - 20 minutes.

The 3DR Solo (referred to as Solo from now on) from 3D Robotics [16] was chosen after surveying at various other options like the DJI Phantom 3 and the Cheerson CX-22 due to its low cost, relatively more supportive development environment, its open source software and its other specifications matching the basic requirements of this project. The 3DR Solo's relevant specifications are listed below in Table 2.1. The 3DR Solo is a well built, durable quadcopter capable of withstanding hard landings, which is perfect for this use case as a gaming device for novice pilots. The quadcopter weighs about 1.7 kg with an attached GoPro HERO4 camera on a fixed gimbal. A fixed gimbal was deemed to be necessary to provide a true mixed reality experience with the user having visual feedback of the drone's movements when the drone pitches or rolls to perform its maneuvers. The fixed gimbal was also necessary to perform reliable vision based localization.

Figure 2.1: 3DR Solo UAV and control joystick.

Table 2.1: 3DR Solo Quadcopter Needs and Specs Table

Need	Specification	Metric
Reasonably compact design	Dimensions (Height; Side length (motor to motor)	10 inches; 18 inches
Last for at least 15 minutes per session	Battery Life (Flight Time)	20 minutes average
Stream low latency HD video wirelessly	Yes, Designed to work with GoPro HERO compact camera	2.4GHz Wifi, 720p HD ; 250ms latency
Accessible software and firmware	Yes, Uses Ardupilot Mega(APM) firmware that can interface with Robot Operating System	-

2.1.2 Sensors on the 3DR Solo

The 3DR Solo comes with a Pixhawk 2 Autopilot running a modified version of open source firmware, Ardupilot Mega Copter(APM Copter). The Pixhawk 2 runs an ARM Cortex-M4 type processor and contains a suite of on-board sensors such as an Inertial Measurement Units (IMUs) that can measure linear acceleration and rate of rotation (roll, pitch and yaw) in the body frame of the Solo. The Solo also contains a magnetometer for measuring cardinal direction and GPS module for latitude and longitude position information, but these two sensors are not available to use for this application due to requirement that the drones have to operate indoors. The magnetometer signal is affected by metal objects and other signals and the GPS, even if the Solo manages a signal lock indoors, does not afford the accuracy needed for this application, as the typical error bandwidth in GPS is in the order of +/- 5 meters. The data from the sensors is input into the APM Copter's algorithms and combined with control inputs to send commands to the Solo's propulsion system.

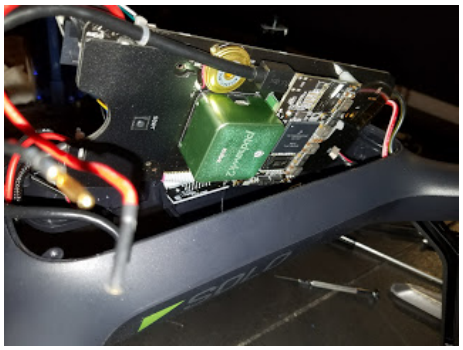


Table 2.2: Left: Pixhawk 2 Autopilot cube with IMU, Right: GoPro HERO 4 camera sensor

A GoPro HERO 4 camera, attached to the Solo by a fixed mount and facing forward, will perform the dual responsibilities of being the main visual sensor for localization and also the coordinate frame in which mixed reality is realized. The streaming video from the GoPro will be the conduit through which the player will be able to see the real world. The GoPro HERO 4 (hereafter referred to as GoPro) is an 'action camera' primarily used for capturing high quality video and storing it on its internal memory card. The

GoPro is also capable of streaming video via its own 2.4 Ghz wireless network or through its HDMI output port. It was determined that the GoPro was a good choice for this application due to the fact that it was supported as an official accessory by the Solo. The Solo contains the internal architecture that can receive the streaming video via the HDMI output port, encode the video and send it to a buffer on Solo controller via the Solo's own 2.4 Ghz wireless network. This was deemed to be the best choice for acquiring the video for sensing and display purposes as there was a risk of the GoPro's wireless interfering with the Solo's wireless, which is important for controlling the drone safely. The relevant specifications of the GoPro is shown in table 2.3.

Table 2.3: Camera Sensor: Go Pro HERO 4 Black Specs Table

Specification	Metric
Weight	90 grams
Battery Life	1 hour average while streaming
Battery charging time	2 hours average
Image data	1280x720 pixels RGB @ 30 Frames/s
Lens type	Wide with 118 deg FOV

The Solo can also stream sensor data from the IMU through it's wireless network, which can be accessed by connecting to the network via a Linux PC and software drivers to parse the data. A limitation of this sensor is that it can only stream IMU data at a relatively low rate of 8 to 10Hz. This is not ideal, but this data can still be used for sensor fusion. The standard deviation of this sensor was experimentally measured to be $0.015 m/s^2$.

2.1.3 Oculus VR headset and gaming setup

The Oculus VR headset was chosen to be the device that would show the player the mixed reality view from the point of the drone. The Oculus VR headset, although primarily used for virtual reality applications, was the best candidate for a headset capable of rendering both the video from the drone and layering the virtual frame of reference containing the game objects onto the real world video view via the Unity graphics engine. The Oculus

VR has an extensive software framework for application developers to take advantage of this engine. The Oculus headset also comes with a pair of joysticks that will be used by the player to navigate the mixed reality landscape by controlling the movement of the Solo.

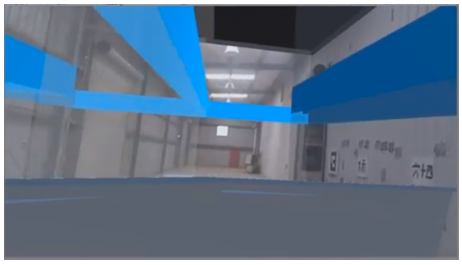


Table 2.4: Oculus VR headset and controllers, Right: View from Headset in our mixed reality maze game

2.1.4 Software setup and functional architecture

The various hardware components of this system needed to be integrated with each other using multiple software packages. The overall project relies heavily on the Robot Operating System (abbreviated ROS, typically pronounced as 'ross'), an open source software package that offers a way to standardize communication between various 'nodes' of a robot, as 'topics' and 'services' carrying 'messages', in order to simplifying implementation and testing. ROS was used as the primary system to route, process and record data between the drone ground station and the graphics processing computer. The drone ground station is a separate Linux machine running the Ubuntu 16.04LTS operating system, which is a requirement for ROS. The graphics processing computer runs the Windows operating system, which is a necessity for running the Unity graphics engine and Oculus software. The functional architecture of the system is shown in Figure 2.2.

The data stream starts with streaming video from the GoPro Camera that includes images of the fiducial markers placed in the real world environment. The movement of the camera view through the environment is controlled by the player via the Oculus joysticks. This is possible by taking the joystick button and axis data, routing it through the Windows machine to the Linux Machine, where the data is converted into a ROS

topic. The joystick data can then be configured to change the movement commands sent to the Solo. The movement commands can be in the form of directional velocity commands or way-point positional commands. The usage of position commands allows the decoupling of the drone movements from the direct joystick control to allow for addition of some movement planning. This is necessary to allow the possibility of adding collision avoidance logic based on the drone's map of the environment. The flow of data through various systems is shown in Figure 2.3.

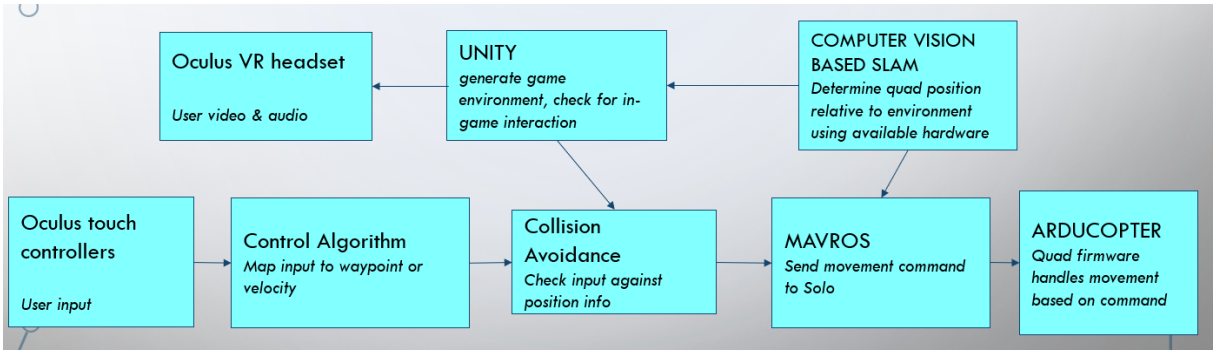


Figure 2.2: Functional Architecture of proposed mixed reality system

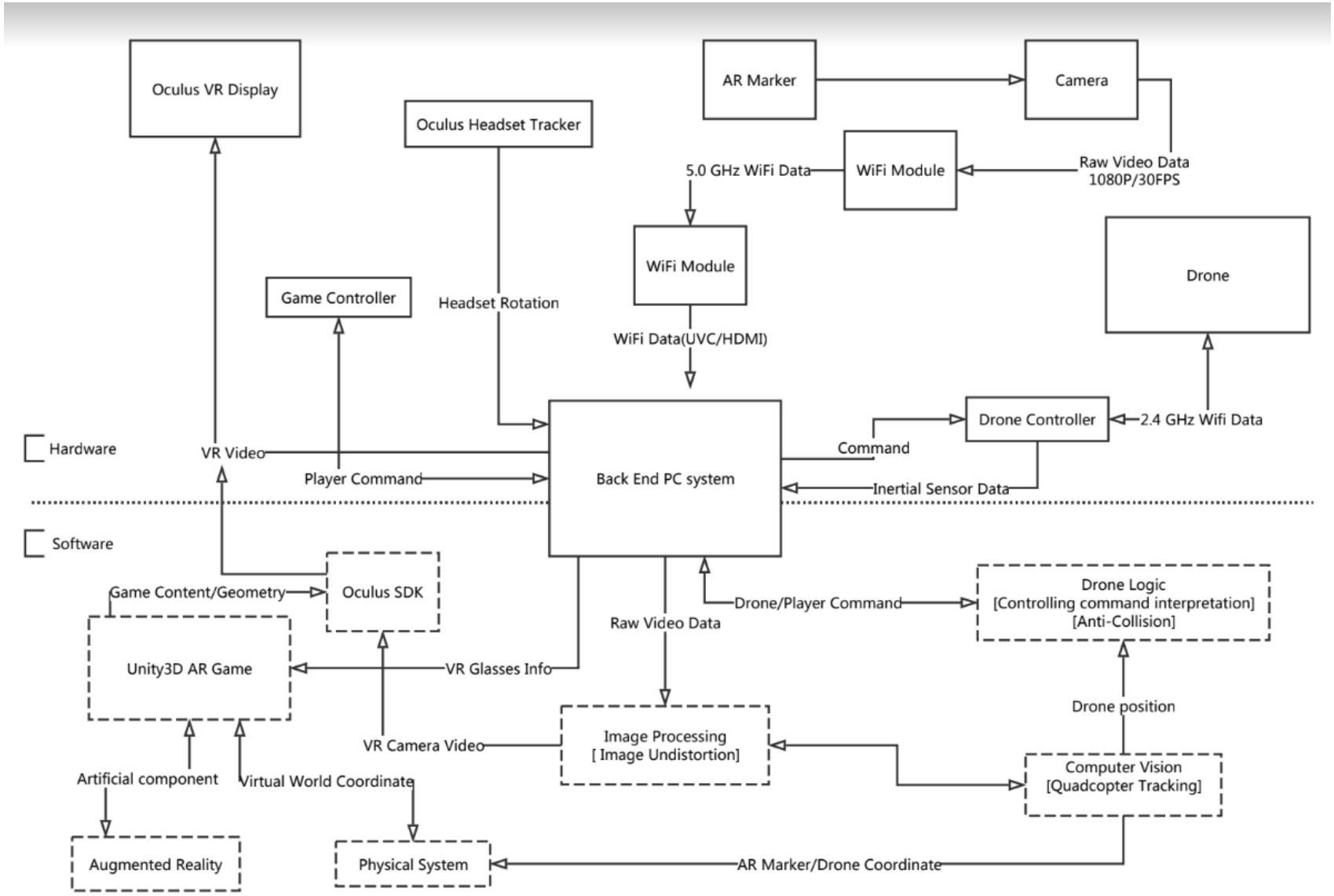


Figure 2.3: Flow of data from various components of proposed mixed reality system

Chapter 3

Pose estimation

Pose estimation or localization is a common problem in robotics where the objective is to obtain the best possible estimate of the pose (position and orientation) of a robot in the world frame of reference. There are many methods to solve this problem, but they can be implemented with interoceptive sensors or exteroceptive sensors or a combination of both. The requirements of this project dictated that the sensors be on board the aerial vehicle so as to minimize the equipment cost of the project. The monocular front facing camera mounted on the Solo for the purpose of providing the user with a view of the real world can also be used to perform pose estimation.

3.1 Monocular Vision Based Pose estimation

3.1.1 Camera Model

The Monocular GoPro camera on the drone is modeled as a pinhole camera. In this model, a view of the world is formed by projecting 3D points onto an image plane through a 'pinhole' that gathers reflected light from the scene. The model is shown in the figure and described by the equation below.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

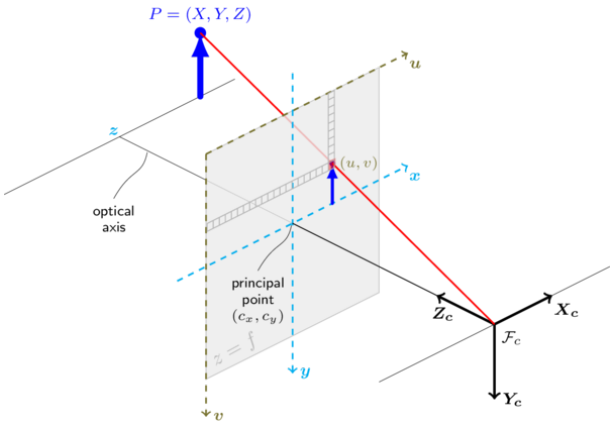


Figure 3.1: Monocular Pose estimation

The equation can be used to describe how the coordinates of a point in the world coordinate space (X, Y, Z) is transformed into a point on the 2D camera space (u, v) (in pixels) scaled by a factor s. The transformation is called a perspective transformation, which is achieved when the input vector is multiplied by an intrinsic camera matrix which captures the intrinsic parameters of the camera like focal lengths f_x, f_y (expressed in pixel units) and location of the principal point (c_x, c_y) and a rotation translation matrix which captures the extrinsic parameters like camera motion around a static scene and transforms the world coordinates to the camera coordinates (x, y, z).The transformation is equivalent to the following when z $\neq 0$.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z \tag{3.2}$$

$$y' = y/z$$

$$u = f_x * x' + c_x$$

$$v = f_y * y' + c_y$$

In order to account for distortion in the lens of the camera, the model is extended to include radial distortion and tangential distortion. The model is extended as follows:

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$
$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_2 x' y' + p_1(r^2 + 2y'^2) \tag{3.3}$$

$$\text{where } r^2 = x'^2 + y'^2$$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

k_1 through k_6 are the radial distortion coefficients and p_1 and p_2 are tangential distortion coefficients. These coefficients determine the degree of distortion in the image. For instance, $k_1 > 0$ indicates barrel distortion and $k_1 < 0$ indicates pin cushion distortion. The distortion coefficients are solely an intrinsic property of a fixed focus camera lens and independent of image resolution.

3.1.2 Method to calibrate camera and obtain intrinsic parameters

The GoPro Hero 4 , was the camera of choice for this project and its intrinsic parameters - $f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2$ were determined using the checkerboard calibration process. Only 3 coefficients of radial distortion (k_1 through k_3) are taken into account in our calibration and the rest are assumed to be zero.

The process employs an object with known geometric pattern and easily detectable feature points like a 2D black and white checkerboard. The calibration process is an algorithm where the steps below were executed for multiple frames with the checkerboard in view, at different positions in the world frame and at different angles:

1. Computes the initial intrinsic parameters using equation (3.3), assuming initial distortion coefficients set at zero.

2. Use the solvePnP [17] algorithm to estimate camera pose.

3. Run the global Levenberg-Marquardt optimization algorithm [18] to minimize the reprojection error i.e the total sum of the squared distances between observed feature points (checkerboard grid corners) and the projected object points using the current

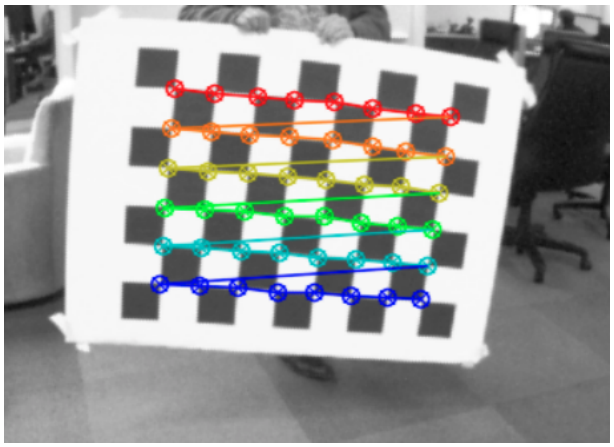


Figure 3.2: Feature points recognized while calibration

estimate for camera parameters and poses.

This calibration procedure was performed using code written in Python with the OpenCV module for initial testing and using the ROS camera_calibration package for subsequent calibration for other GoPro cameras. The final camera parameters for a typical GoPro Hero 4 were obtained and resulting camera matrix and the distortion matrix are shown in equation (3.4)

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 593.134 & 0 & 641.997 \\ 0 & 597.456 & 343.931 \\ 0 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} k_1 & k_2 & k_3 & p_1 & p_2 \end{bmatrix} = \begin{bmatrix} -0.225 & 0.040 & 0.0 & 0.002 & 0.001 \end{bmatrix} \tag{3.4}$$

3.1.3 Perspective and point method for Pose estimation with ArUco Markers

The camera parameters obtained in the previous step and the solvePnP algorithm which requires features recognized reliably in the world coordinate space, is used with coded fiducial markers from the OpenCV [19]. The detection of square fiducial markers called ArUco is implemented within the OpenCV library based on work by Rafael Muoz and Sergio Garrido [20].

The OpenCV ArUco library provides algorithms that help with marker detection (i. e.,

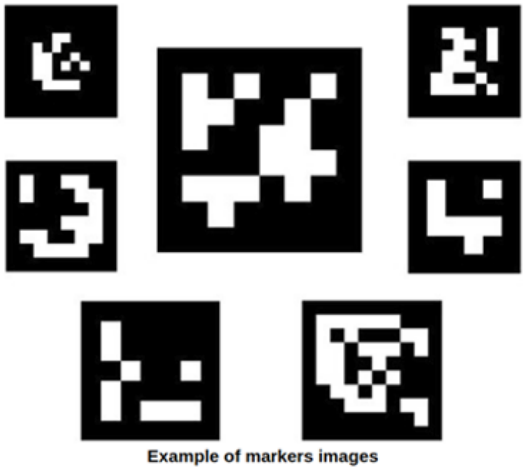


Figure 3.3: Coded Fiducial markers - Aruco type

obtaining image points and marker id via a bits extraction method) and pose estimation using the solvePnP method. The solvePnP method iteratively solves for the variables R and t (rotation matrix and translation vector) based on the distorted camera model (3.2) and (3.3) given the image points, camera parameters (3.4), and object points using the Direct Linear Transform method followed by the Levenberg-Marquardt optimization to minimize the re-projection error. This gives the output rotation vector and translation vector of the marker with respect to the camera in the world coordinate space. If we specify the initial position of the camera in the world coordinate space, we can now obtain the pose of the camera with respect to a fixed marker.

3.1.4 Experimental Setup

The experiment for pose estimation was setup in the Cyber Human Physical System (CHPS) lab with the GoPro camera attached to the Solo drone via a fixed mount. The CHPS lab offered the opportunity to test the system in controlled lighting and no-wind conditions, since worse conditions may disrupt the efficacy of the vision based system. Also, the lab's test area (shown in figure 3.4) is equipped with OptiTrack, a state of the art indoor positioning system based on motion capture technology capable of millimeter level accuracy. The OptiTrack system was configured to provide a ground truth pose estimate for comparison. The special spherical reflective markers used by OptiTrack were attached to the Solo using sticky tape and the markers were configured as a rigid body.

The OptiTrack system is able to provide the 6DOF pose of this rigid body with its body fixed coordinate ***optitrack_frame*** in the ***world_frame*** which is preset to be the center of the testing area.

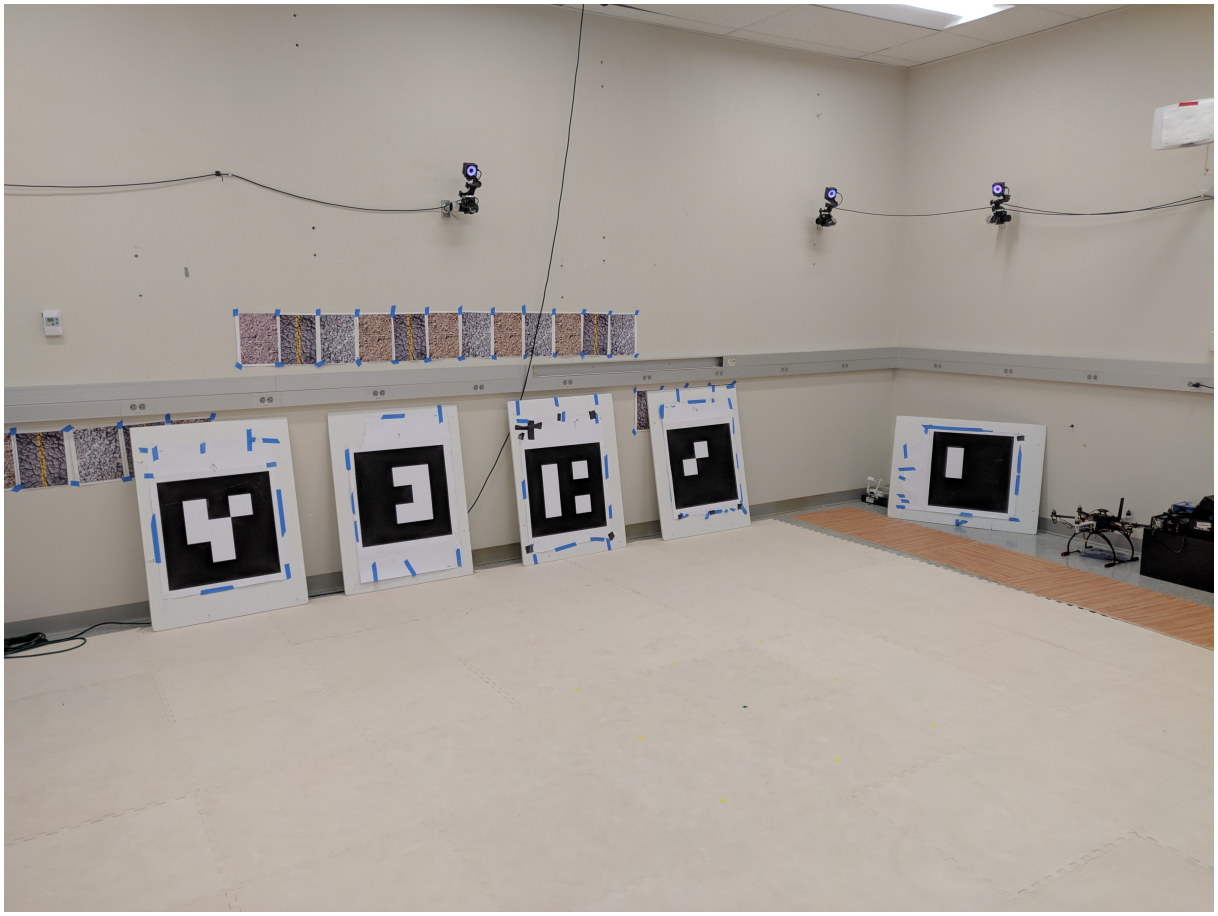


Figure 3.4: Test Area at CHPS Lab

The Solo's center of mass ***base_frame*** and its main sensor, the GoPro camera ***camera_frame*** are each given their own body fixed coordinates. The coordinate transformation between ***base_frame*** and the ***camera_frame*** can be known via measurement. The other coordinate systems needed to superimposed on each other in order to compare the measured values of the ground truth and the vision estimate. The coordinate systems were setup as shown in figure 3.5.

The camera view (figure 3.6 (d)) shows the real-time image that is being streamed from the GoPro via the Solo's Wifi connection. The images are received into the computer via a video capture card and parsed at the rate of 30Hz (frames per second). The 'cv_camera'

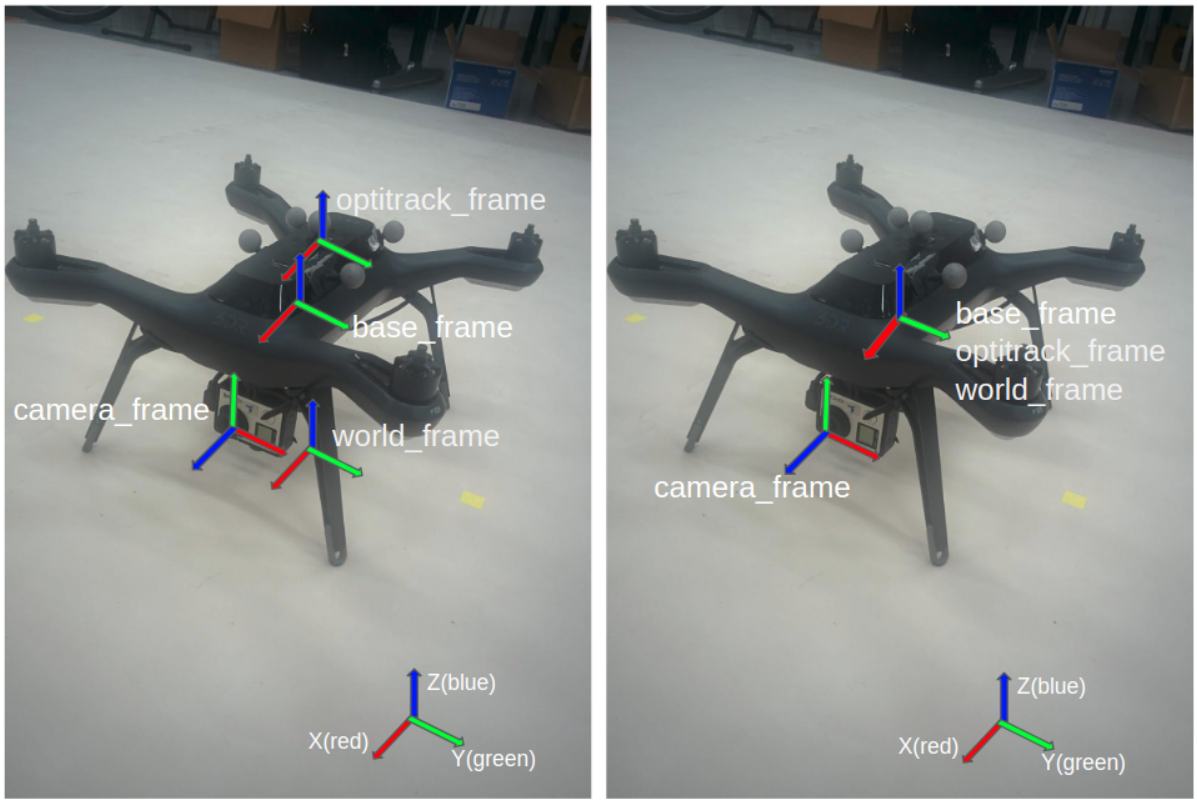


Figure 3.5: Left:Separate coordinate systems before setup , Right: After transformation, the other frames now overlap with the base_frame

ROS package was used to read the USB output of the video capture card and convert it into meaningful image data in the form of a ROS topic. The image data and the camera parameters were used as inputs to the solvePnP algorithm. The marker detection and solvePnP algorithms were implemented for real-time usage with the 'aruco_detect' ROS package. The parameters for marker detection and solvePnP algorithm were tuned for optimal speed and accuracy. A single ArUco marker with a known marker area is placed in the range of the camera sensor.

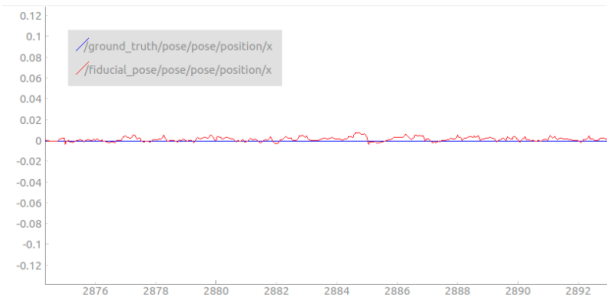
3.1.5 Results and Discussion

The results of the monocular vision based pose estimation using a single ArUco marker is depicted in the figures of this section. The first array of images in figure 3.6 depict the sensor values for the position of the drone (***base_frame***) in the center of the lab test area (***world_frame***). The translation value of the vision estimate (red line) can be compared

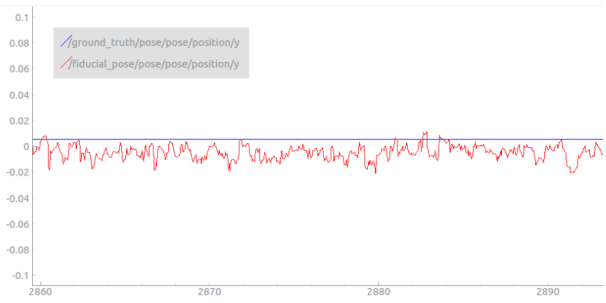
with the OptiTrack sensor values (blue line), which will be considered the ground truth in all the tests. The stationary test conveys an good idea of the visual sensor's noise and its variance. It was observed that the x-position values showed the lowest variance. This is explained by the fact that solvePnP algorithm able to minimize the re-projection error in the drone's X axis (Z axis in the ***camera_frame***) due to its direct relationship with the size of the marker area in the image frame.

The second array of images in figure 3.7 shows the results when the drone is moving in the ***world_frame***. The drone was moved specifically in a '3D cross' trajectory in order to capture the change in the values of the translation vector independently. The z position is changed first, as the drone lifts off the ground to a height of about 1.55 meters, very close to camera sensor losing sight of the marker, then back down to a stable altitude of 1 meter and then the x-position is perturbed in the positive and negative direction, followed by a change in the y-position. It was observed that the x-position value tracked relatively well with the ground truth value, with a maximum error value of 0.1m when the fiducial marker was around 3m away from the drone. Most of the error value can be attributed to the delay in sensor signal travelling via WiFi after various encoding and decoding processes, in addition to the off-board processing of the pose estimation algorithm. The y-position value tracked the same way during the motion, but noise is observed in the y-position during the large change in position in the x and z axes. Similarly, the z position is greatly affected by noise when the marker is close the edge of the camera sensor's range.

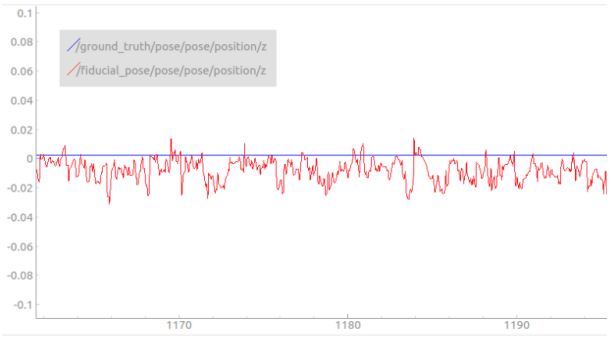
The third image array 3.8 gives an idea of the overall tracking performance while the drone is moved by hand in an unnatural circular motion. The z- position performance is affected greatly when the drone is far away (maximum x-position) from the marker. In the current setup with a single marker, it is evident that the drone cannot be yawed freely as it would lose sight of the fiducial marker. This problem is addressed by implementing the SLAM algorithm described in the next chapter.



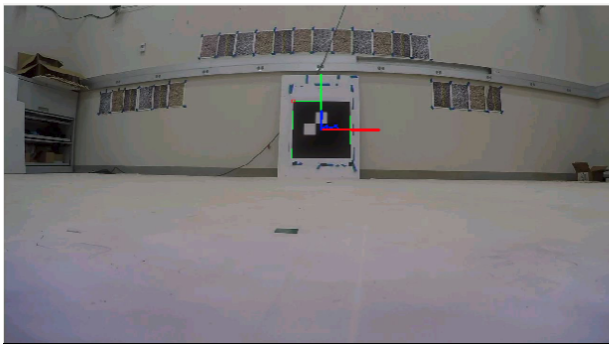
(a) x-position (σ^2 =2.5×10^{-5} m^2)



(b) y-position (σ^2 =1×10^{-4} m^2)

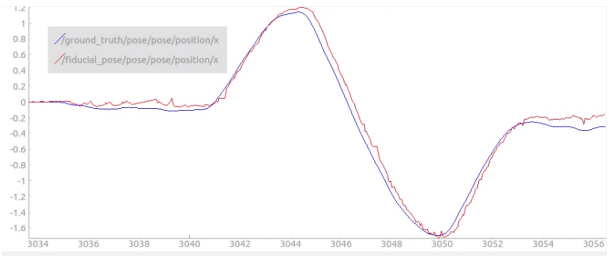


(c) z-position (σ^2 =2.3×10^{-4} m^2)

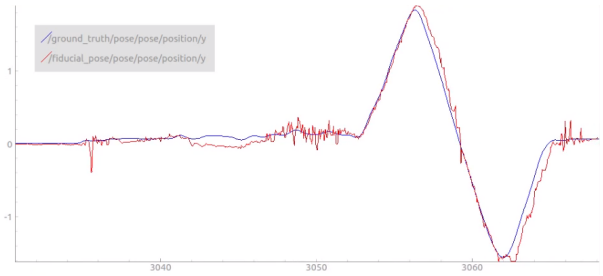


(d) Image view with recognized marker

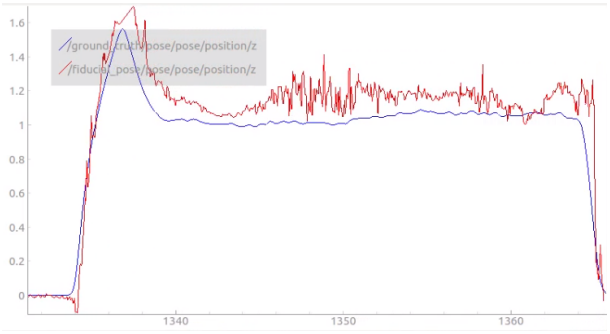
Figure 3.6: Vision sensor pose estimate compared to ground truth while drone is stationary. All position variables are in meters and time series in seconds.



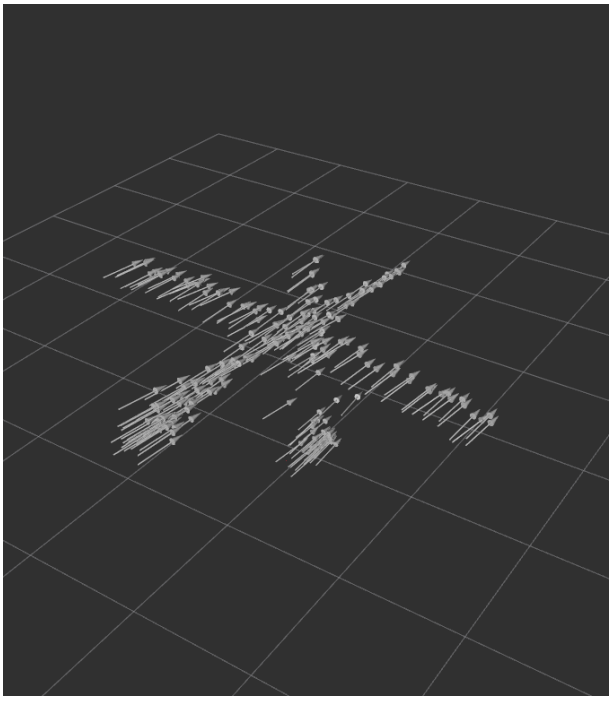
(a) x-position (meters) over time (seconds)



(b) y-position

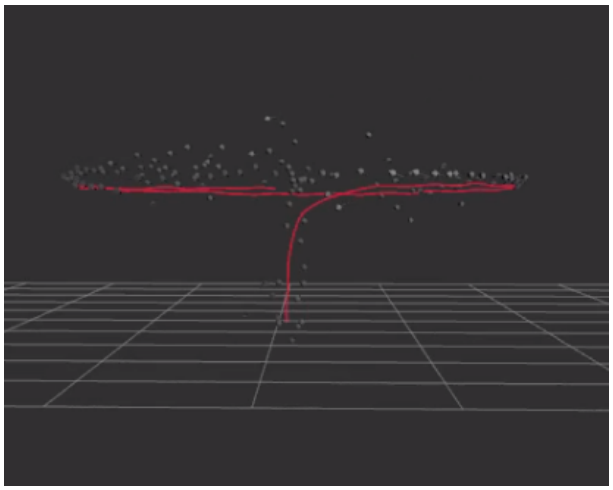


(c) z-position

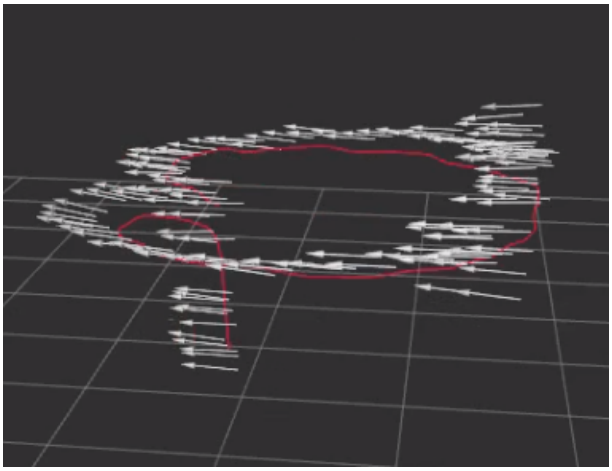


(d) XYZ Motion Visualization

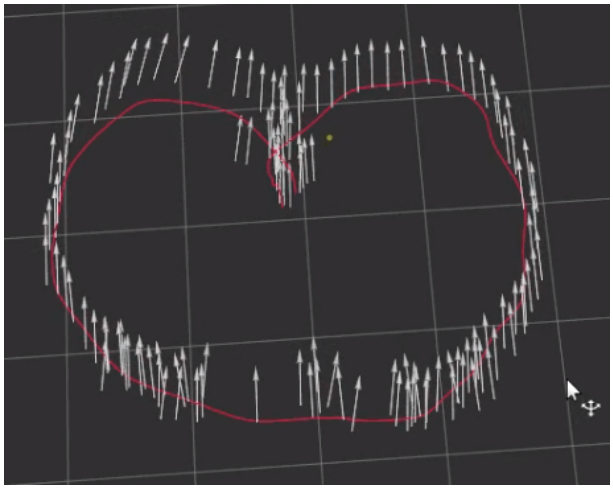
Figure 3.7: Vision sensor pose estimate compared to ground truth while drone is moved in a '3D Cross' motion visualized in (d). Orientation and location of the white arrows represent the pose estimates.



(a) Front View



(b) Side View



Top view

Figure 3.8: Vision sensor pose estimate compared to ground truth while drone is tracked in a curved motion.

Chapter 4

Simultaneous Localization and Mapping (SLAM)

Simultaneous localization and Mapping or 'SLAM' is a fundamental problem in robotics, where the sensor measurements from a robot are used to construct a map of an unknown environment while simultaneously keeping track of the robot's pose within that map. SLAM is a ongoing hot research topic in the robotics field, especially in the market of trying to build the first fully autonomous cars. According to [21], current maturity of the effective solutions to the SLAM problem is highly dependent on following aspects of the robot:

1. Type of motion: dynamics, maximum speed, available sensors, their resolution and sampling rate and computational resources.

2. Environment: whether planar or three dimensional, presence of landmarks, amount of dynamic elements, degree of symmetry and risk of perceptual aliasing.

3. Performance requirements: Desired accuracy in the estimation of the state of the robot, type of representation of environment, estimation latency, success rate and maximum size of mapped area etc.

In earlier implementations of SLAM for robotics, there was a need to integrate various types of sensors such as motor encoders, laser range sensors, inertial measurement sensors, GPS and cameras. SLAM with only cameras is a much harder problem, but in recent years, with the large gains in the processing capabilities of portable chipsets for robotics,

there have been some advances in visual SLAM (vSLAM) techniques. vSLAM techniques can be categorized into feature-based approaches and direct approaches. Feature based approaches track and map feature points in an environment. The number of features correlates directly with the efficacy of the SLAM algorithm.The direct approach tries to estimate motion by tracking the image as a whole, so it can work in texture-less or feature-less environments [22].

Some of the feature based methods used in the past are MonoSLAM [23], in which camera motion and the structure of the unknown environment is are simultaneously estimated using a tightly coupled Extended Kalman Filter (EKF). The drawback of this method is that it has a very high computational cost.

To overcome the drawback of MonoSLAM, Parallel Tracking and Mapping (PTAM) [24] was proposed by Klien and Murray. PTAM splits the tracking and mapping into different parallel CPU threads, so that the computational cost of the mapping does not affect the tracking. PTAM uses the Bundle Adjustment (BA) optimization technique instead of a tightly coupled EKF, giving it the ability to handle a large amount of feature points. Marker based SLAM is also a type of feature based approach that was chosen for this project along with pose-graph optimization, due to the difficulty in finding feature points in the test environments. Direct approach methods like DTAM and LSD-SLAM were not used due to the relatively high computational cost, and unsuitability for large environments.

4.1 Marker Based SLAM

SLAM is a requirement in this application because the robot with its sensors needs to be able not only be able to localize itself with respect to a single fiducial marker, but have the freedom to pitch, roll and yaw while moving, requiring multiple markers placed all over the game environment. The final game environment will be fixed and known, but for testing, it was determined that the fiducials could not be reliably fixed to the environment for every test, so a map of fiducials needed to be generated by the SLAM algorithm, instead of only performing localization. The primary sensor in this case is the monocular wide

angle lens GoPro camera streaming image data at 30Hz. The SLAM algorithm needed to work off-board the drone on a ground station due to computational restrictions, so there was expected to be latency involved in acquiring sensor data and sending control inputs to the drone in order to control the dynamics. The environment would be a cuboid with known dimensions, with the possibility of additional virtual or physical obstacles in the space. The performance of the SLAM needed to be fairly accurate (in the range of 5 - 10 cm) in order to perform dynamic collision detection and avoidance and as accurate and smooth as possible to improve the fidelity of the mixed reality simulation.

4.2 Pose graph Optimization

The pose graph is a type of SLAM optimization where the variables to be estimated are poses sampled along the trajectory of the robot and each 'factor' in a graph of factors imposes a constraint on a pair of poses. The different factors could be a prior estimate or zero point, camera observations in relation to the camera's intrinsic parameters, odometry constraints, loop closure methods, or a combination of all of the above.

In our case, the main factors are the prior map of markers or no map if the system has not been initialized yet, the camera intrinsic parameters re-projected from the image points to the world points to check re-projection error, and some loop closure conditions to build a consistent map of the marker locations in the environment.

Figure 4.1 shows a visualization of the pose graph SLAM with the $x_1...x_n$ representing the 6DOF camera pose variables to be estimated per time step n, p representing the prior map of poses as a constraint, along with loop closure constraints $c_1...c_n$ and camera observation parameters $v_1...v_n$ in relation to camera intrinsic parameter constant reference K. The pose of the markers with respect to current camera pose is stored in variables $l_1...l_n$.

The calculation of pose of the drone from a single marker is used the starting variable and its observation and variance calculated from the re-projection error are used as constraints to calculate the pose variable for the next time steps. The vertices of every marker in the sensor image data are detected, identified as an unique marker based on the bit code and its pose with respect to the world frame is calculated. This information

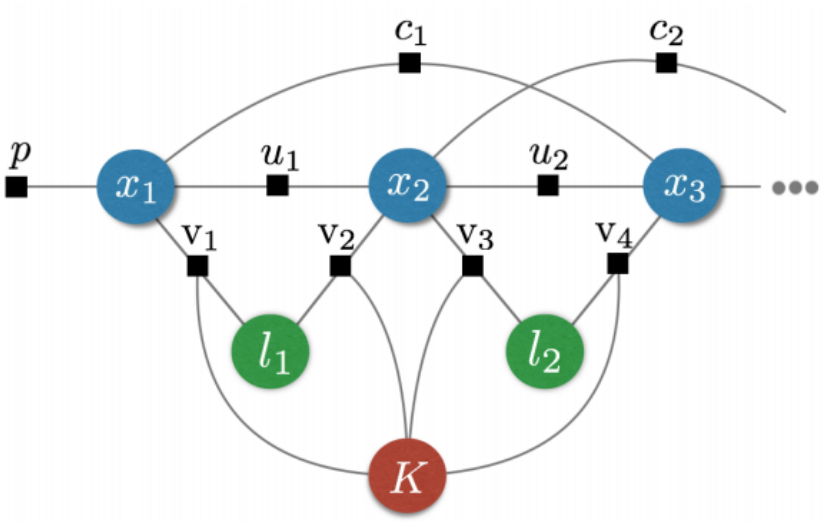


Figure 4.1: Typical Structure of Pose graph SLAM [21]

is stored in map information file as a constant value. The observation constraint checks the camera pose in relation with the observed markers in the map and uses the lowest variance measurement in conjunction with the camera parameters to estimate the drone's pose in the world_frame. Figure 4.2 shows the flow of data as implemented in this Marker based Visual SLAM algorithm.

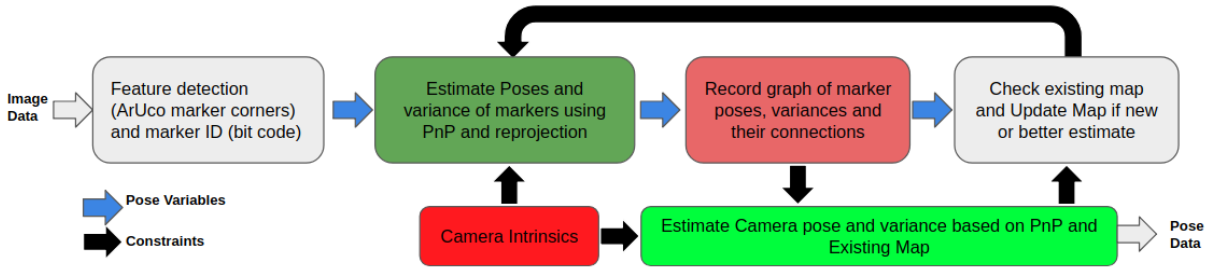


Figure 4.2: Marker SLAM data flow

4.3 Experimental Setup and method

Six Aruco markers with 3×3 bits and one border bit were made and printed out on large 0.55 meter square sheets in order to ensure detection over large distances. For the first test, the markers are placed in view of the camera sensor, such that changes in yaw angle

(up to +/-90 degrees) or translation in the X or Y direction would still ensure that at least one marker is visible at all times as shown in figure 4.3.

The implementation of pose graph SLAM was carried out using a modified version of fiducial_slam ROS package. The map of poses is initialized only when the algorithm starts, in order to allow the SLAM algorithm to generate it's own map of markers. The zero point pose estimate was calibrated to coincide with the center of the test environment and ground truth zero measurement.

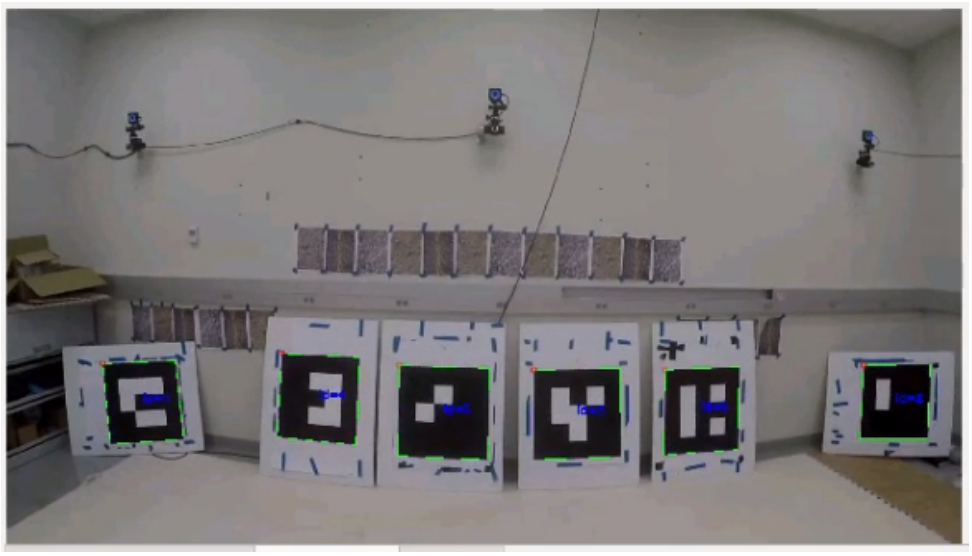
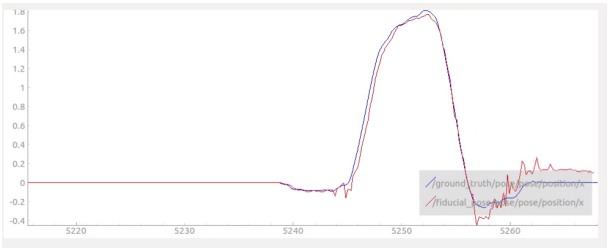


Figure 4.3: SLAM marker placement for XYZ test

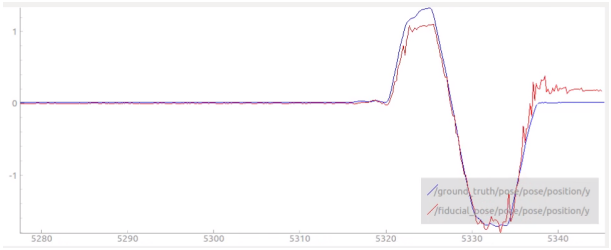
4.4 Results and limitations

The results of the SLAM algorithm using multiple ArUco markers is depicted in the figures of this section. The first array of images in figure 4.4 shows the results of the SLAM position tracking test. This test is designed to mimic a scenario in a mixed reality 3D shooting game. The drone is moved in an approximately rectangular path such that different markers appear in and out of view at different times. Even with wide 118 degree field of view of the camera, the some of the markers are lost from view when the drone gets close to the marker 'wall' as shown in figure 4.4 (d). The tracking performance compared to ground truth is seen in 4.4 (a),(b) and (c) with the blue line representing ground truth and the red line, the SLAM pose estimate. The x position tracks very well, similar to the

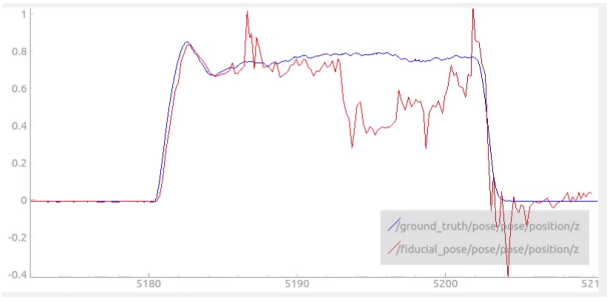
single marker test, and the y position performs well until the drone reaches top corners of the rectangle, close to the diagonally placed markers and some accuracy is lost due to the large area of the marker features being close the edges of the image. This is the area where there is large re-projection error due to high distortion. The distortion model is not perfect and is being approximated up to the second order as described in Chapter 3. The Z position tracks well when the drone moves upwards off the ground, but suffers a lot of noise and loss in tracking accuracy when the markers are very close and all the feature points (marker corners) are in the heavily distorted areas of the image, where the re-projection error tends to be the highest.



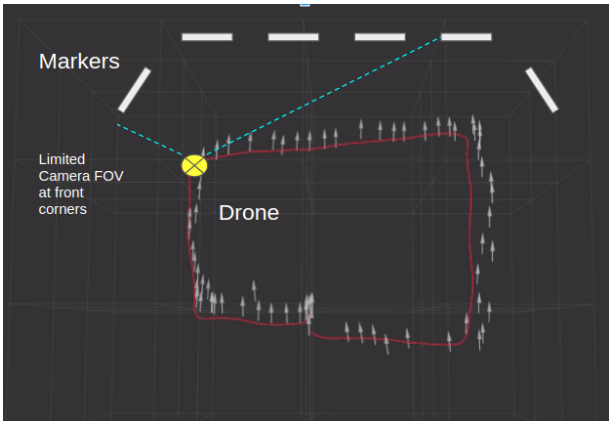
(a) x-position



(b) y-position



(c) z-position

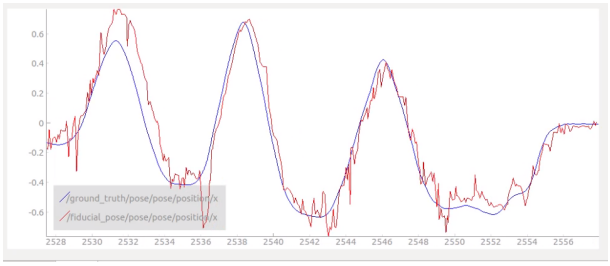


(d) SLAM marker placement for square path test

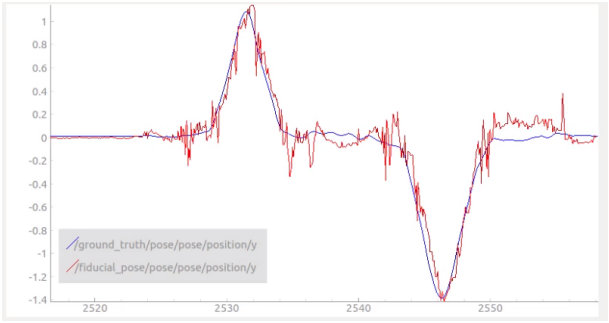
Figure 4.4: Vision sensor pose estimate with SLAM recognizing multiple markers compared to ground truth while drone is moved in a square path. Orientation and location of the white arrows represent the pose estimates.

The second test involved moving the drone in a 'V' shaped trajectory by starting from the center, performing a yaw to about 45 degrees to the left, and going forward to the edge

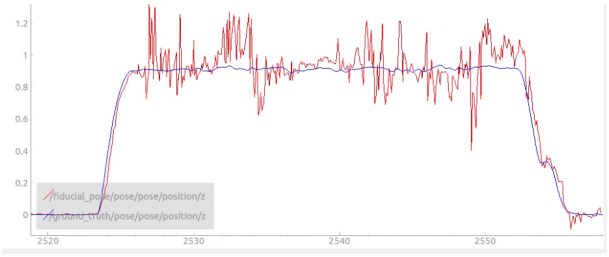
of the room and then back to the center, yaw about 90 degrees to the right and forward towards the right edge of the room. This test is simply to evaluate position tracking after a yaw operation is performed, and the drone loses sight of some of the markers. It was observed (Figure 4.5 (a),(b) and (c)) that there is a lot of noise created in the position while yawing the drone and moving in a diagonal direction, but generally the position tracking was still available. It is possible to reduce the levels by adding more markers and not getting as close to the marker.



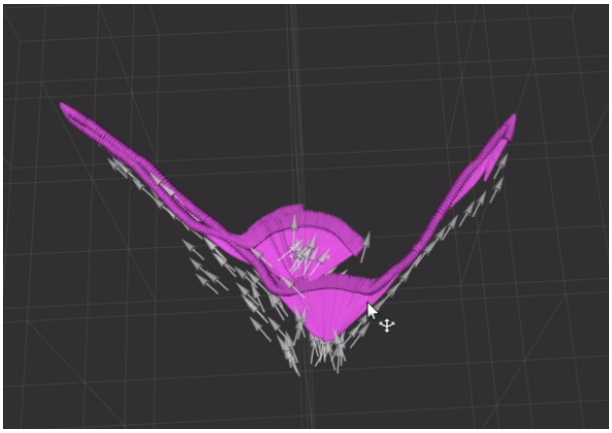
(a) x-position (meters) over time (seconds)



(b) y-position



(c) z-position

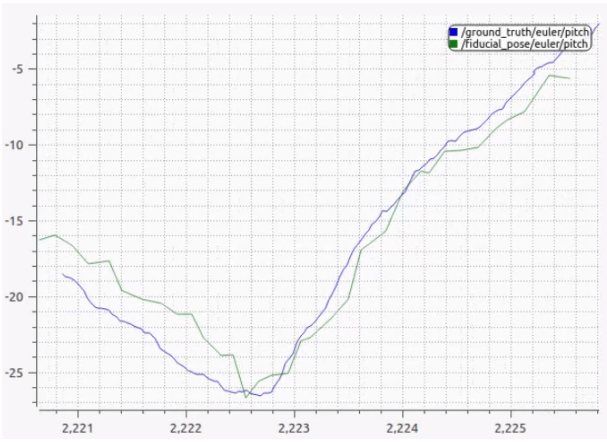


(d) SLAM Pose Estimate plot of 'V' path.

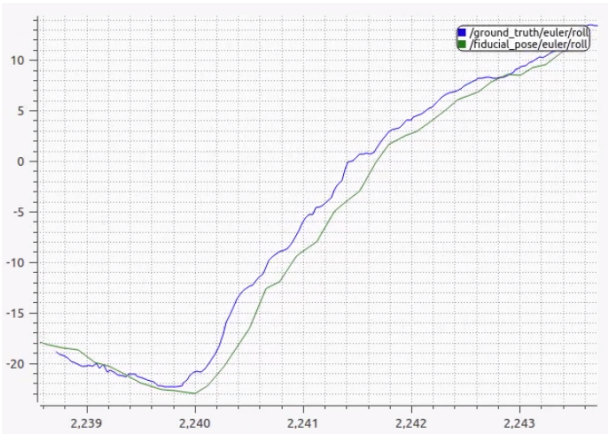
Figure 4.5: Vision sensor pose estimate compared to ground truth while drone is moved in a 'V' path. Orientation and location of the white arrows in (d) represent the pose estimates. The magenta arrows, seen overlapping to create a surface-like appearance, represent a high resolution ground truth estimate of the maneuver.

The final test for SLAM is the ability to track orientation when the drone is moving. The results in figure 4.6 show the change in Euler angles when the drone is moved in the

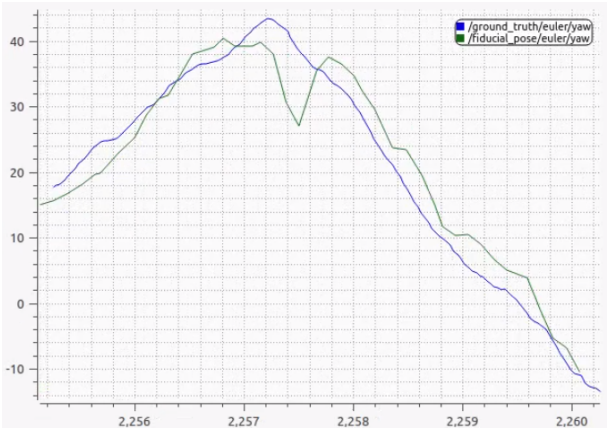
3D environment. The SLAM system is able to pick up sudden changes in motion in pitch and roll cases but fast yawing motion results in inaccurate tracking.



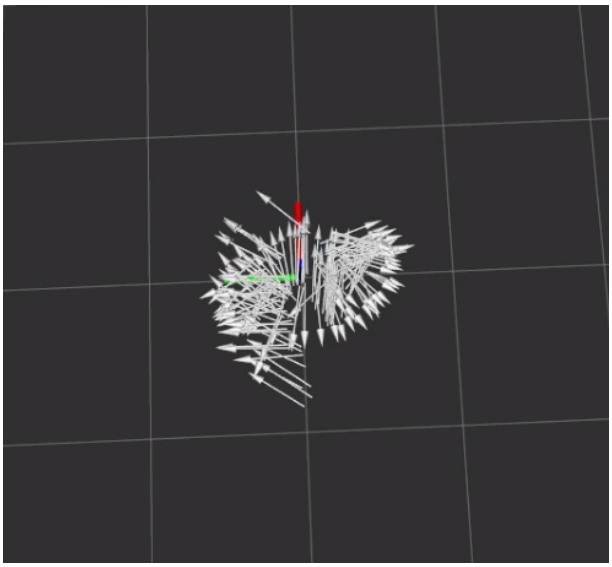
(a) Roll (degrees) over time (seconds)



(b) Pitch



(c) Yaw



(d) Full Z rotation test.

Figure 4.6: SLAM pose estimate of drone orientation represented by Euler angles compared to ground truth while drone is moving or yawing. Orientation and location of the white arrows in (d) represent the pose estimates.

Chapter 5

Sensor Fusion using Extended Kalman Filter

Using a single visual sensor clearly seemed to have disadvantages in with respect to scaling, especially in a large environment and non ideally placed markers. Noise is also an issue due to the fact the visual sensor is adversely affected by sudden movements causing pixel distortion or blurring, losses in transmission or artifacts and light conditions affecting marker visibility and feature point detection. Also, pose estimation with a monocular camera in general also performs badly when the feature points are close to the edge of the image.

To address these issues and limitations of the single visual sensor, it was determined that information from the additional existing IMU sensor could be used to complement the visual sensor and arrive at a more robust estimate that is more sensitive to the short term changes in the pose of the drone. The IMU sensor by itself is also ill suited for pose estimation due to the susceptibility to drift caused by double integration errors in the accelerometer measurements.

Corke et al [25] describes two broad approaches to combining inertial and visual measurements by sensor fusion: loosely coupled and tightly coupled. The loosely coupled philosophy involves treating both sensors as separate modules running at different rates and fusing information externally, while the tightly coupled approach combines both information threads into a single optimal filter at an iteration level. Weiss and Siegwart [26]

use a loosely coupled type sensor fusion method to work on-board a Micro Aerial Vehicle (MAV) with a 1.6 Ghz Atom processor with some success. Tightly coupled approaches such as the one implemented by Jones and Soatto [27] are generally more computationally intensive and less modular, but offer better accuracy.

Sensor Fusion is typically performed either Kalman Filter (KF), the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) or a Particle filter. According to [28], [29], [30], the EKF technique was determined to be the most suitable for visual-inertial fusion methods. UKF is also computationally more intensive than EKF.

In this project, the camera sensor's data is first processed using an image processing algorithm to determine the pose (position and orientation) of the camera, and IMU sensor readings are acquired relative to the base_link of the drone. The pose of the camera is transformed to the base_link coordinate frame to match with the IMU. Both these measurements are now sent to the EKF filter, with knowledge of their co-variances. This setup is now a loosely-coupled EKF as illustrated in figure 5.1

The existing IMU sensor in our Solo platform as described in chapter 2 is only able to transmit at around 10Hz to the off-board ground station where the sensor fusion is performed, so it is anticipated that the sensor fusion will not be able to perform to its best capability, but the results of this implementation will allow us to get a baseline for the problems that the sensor fusion will solve. The coordinate systems for the various sensors used are described in Chapter 3.

5.1 EKF model for sensor fusion

The EKF is the non-linear version of the popular Kalman Filter, used widely in state estimation and control applications. The EKF is primarily used to optimally estimate the state of any non-linear stochastic process given the knowledge of its noise characteristics, measurements of the state variables by any sensors and the sensors' noise characteristics. Our goal is to estimate the full 3D (6DOF) pose and velocity of the drone over time. The non-linear system can be described by:

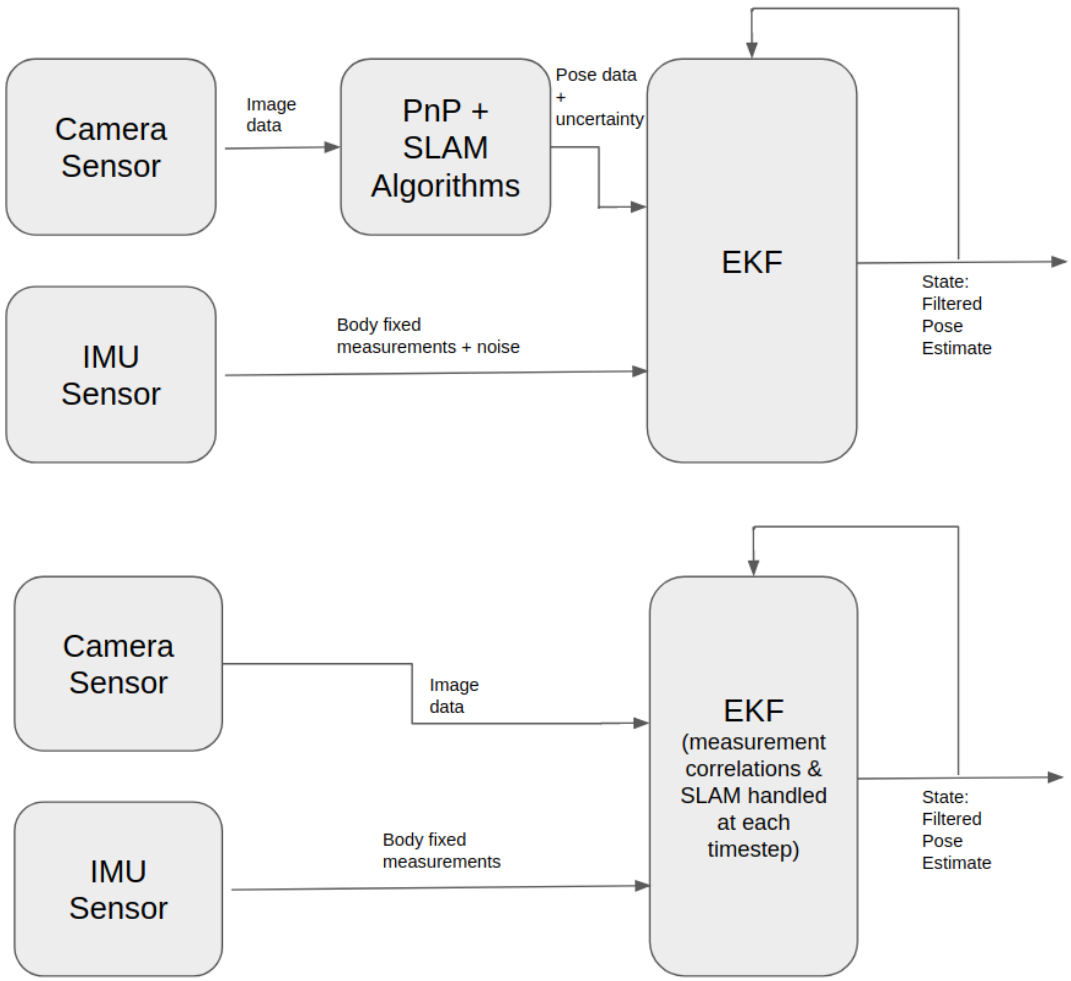


Figure 5.1: Loosely (top) and tightly (beneath) coupled EKF solutions

$$x_k = f(x_{k-1}) + w_{k-1} \tag{5.1}$$

where x_k is the drone's 3D pose at time k, f is a non-linear state transition function and w_{k-1} is the process noise, assumed to be normally distributed. The state vector is 12 dimensional, containing, the drone's 3D pose, 3D orientation and the rotational and translational velocities. The non linear state transition model is for an omni-directional robot. The state transitions of the pose and velocity elements are kinematic. For example:

$$x_k = x_{k-1} + \dot{x_k}\Delta t + 0.5\ddot{x_k}\Delta t^2 \tag{5.2}$$

and

$$\dot{x_k} = \dot{x_{k-1}} + \ddot{x_k}\Delta t \tag{5.3}$$

The equation becomes non linear when the measured accelerations received in the body frame ***(base_frame)*** have to transformed into the inertial frame ***(world_frame)*** by using the transformation matrix as shown in equation (5.4)

$$\begin{bmatrix} x_{wf} \\ y_{wf} \\ z_{wf} \end{bmatrix} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} x_{bf} \\ y_{bf} \\ z_{bf} \end{bmatrix} \tag{5.4}$$

where (x_{wf}, y_{wf}, z_{wf}) and (x_{bf}, y_{bf}, z_{bf}) are the world frame and body frame kinematic variables (could be position, velocity and acceleration), (ϕ, θ, ψ) are the Euler angles of rotation (roll, pitch and yaw) about the body frame and s is an abbreviation for *sine*, c for *cosine*. In our case, the linear acceleration variable is measured in the body frame with the IMU and will be converted before integrating in the state transition model. The entire state transition model including the linear acceleration terms can be modeled as a large 15×15 matrix, but only the first 12 terms (excluding the linear accelerations) are carried over the next step. The accelerations are converted into velocities using (5.3).

The sensor measurements received are in the form of equation (5.5)

$$z_k = h(x_k) + v_k \tag{5.5}$$

where z_k is the sensor measurement at time k, h is a non-linear sensor model that consists of a matrix with rows being the measurement values from each sensor and columns being the full state variables and ones in the location of the values to be updated. This model maps the state into the measurement space directly as this is a kinematic model. v_k is the measurement noise, assumed to be normally distributed.

The Extended Kalman Filter algorithm is typically performed with a prediction step and correction step cycle as described in the equations below. The prediction step (equations (5.6) and (5.7) propagates the state estimate and error co-variance forward in time.

$$\hat{x_k} = f(x_{k-1}) \tag{5.6}$$

$$\hat{P}_k = FP_{k-1}F^T + Q \tag{5.7}$$

The estimated error co-variance P is projected forward via F which is the Jacobian of f, and then perturbed by the process noise co-variance Q. The Jacobian is necessary to approximate the non linear function partially with respect to each state variable at each time step k. The correction step is described in equations (5.8),(5.9), and (5.10).

$$K = \hat{P}_k H^T (H\hat{P}_k H^T + R)^{-1} \tag{5.8}$$

$$x_k = \hat{x_k} + K(z - H\hat{x_k}) \tag{5.9}$$

$$P_k = (I - KH)\hat{P}_k(I - KH)^T + KRK^T \tag{5.10}$$

The observation matrix H, the measurement co-variance R and P_k are used to calculate the Kalman gain K. Since each sensor measures the kinematic state directly (bias correction is handled separately), H is an identity matrix. The Gain is used to update the state vector and the co-variance matrix. Equation (5.10) uses the Joseph's form to ensure that P_k is positive semi-definite. The running estimate of the state vector is the optimal state observer that can minimize the propagated error estimate over time.

5.2 Implementation of EKF

For sensor fusion of measurement values from two different sensors measuring different parts of the state vector i.e, the IMU measuring linear acceleration $(\dot{v_x}, \dot{v_y}, \dot{v_z})$ and rotational velocity $(\dot{\phi}, \dot{\theta}, \dot{\psi})$, and the vision sensor measuring position (x, y, z) and orientation (ϕ, θ, ψ) are used to populate the measurement vector z in equation (5.9). The acceleration values are integrated at each time step to obtain the velocity, which is part of the state vector. The unmeasured terms in the vector are assumed to be zero.

The loosely coupled Extended Kalman filter for this project was implemented in ROS with the help of robot_localization package developed by Charles River Analytics [31] which provides the framework for high speed computation of the EKF algorithm in real-time.

The measurement co-variance matrix R is setup with the diagonal elements corresponding to the two different sensors i.e, the sensor's variance σ^2 with respect to itself. These values for the visual sensor, corresponding to position in the R matrix, is calculated in the solvePnP algorithm for each time step k as the minimized re projection error for each image frame. The values for the IMU sensor, corresponding to the linear acceleration and rotational velocity, is determined experimentally by calculating the variance over a sample range of measured values. The variance of linear accelerations ${\sigma_a}^2$ was calculated to be 0.005 $(m \cdot s^{-2})^2$ and the variance for rotational velocity $\sigma^2_{\dot{\phi},\dot{\theta},\dot{\psi}}$ was calculated to be 0.001 $(rad \cdot s^{-1})^2$.

The Process co-variance matrix Q is a 15×15 matrix that is added to linear approximation of the state error in time step $k-1$ to obtain the estimated state error in the current time step as per equation (5.7). This co-variance matrix needed to be tuned with respect to each state vector. The obtained manually tuned value for this fusion application is shown in equation (5.11). The initial value for state error estimate P_k was also obtained experimentally by observing the performance of the EKF. They are both diagonal matrices, with only the self correlation values of the state vector estimated.

$$\begin{aligned} Q &= \text{diag}(x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, \ddot{x}, \ddot{y}, \ddot{z}) \\ &= \text{diag}(1 \times 10^{-1}, 1 \times 10^{-1}, 1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-2}, 1 \times 10^{-2}, 1 \times 10^{-3}, \\ &1 \times 10^{-3}, 1 \times 10^{-3}, 1 \times 10^{-3}, 1 \times 10^{-3}, 1 \times 10^{-3}, 1 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-5}) \end{aligned} \tag{5.11}$$

5.3 Parameters considered while performing EKF Sensor fusion

There are a number of tunable parameters that affect the final state estimate of the loosely coupled EKF algorithm.

The ***frequency*** at which the filter is run is an important parameter that can affect the behaviour of this system. Higher frequencies require more computational resources, but can allow the filter output to be more incremental and smooth. But, bad data can easily cause the filter to lose tracking for many time steps until the error causes the filter to converge to the model. The SLAM estimate runs at around 20 Hz and the IMU at around 10Hz. The filter waits for updates based on the sensor timeout setting and the new measurements are added to the z vector term in the equation (5.9) if available. If only one or no measurement is available, the correction step is skipped and the results of the prediction step is used to propagate the filter estimate to the next time step. The filter itself can be theoretically run at any frequency, but a frequency of 10Hz was used in the results of the tests.

Sensor timeout (in seconds) is another parameter that is useful in conjunction with the frequency that allows the ability to control whether the filter runs the correction step after the prediction step after measurement is received. ***Queue size*** of the data to be fused determines whether the old measurement are fused after waiting for the prediction step to complete.

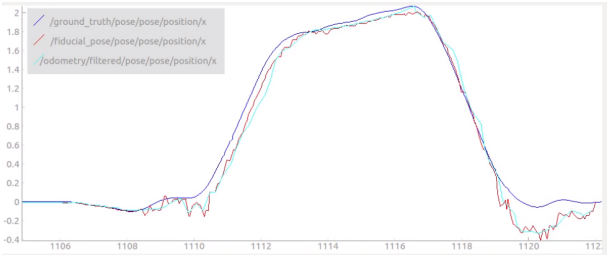
IMU calibration was performed before every test, by letting the IMU bias term converge in the EKF and also by removal of large constant biases, such as the acceleration in the downwards Z direction due to gravity.

Zero calibration of the three coordinate systems was performed before every test. But since the minimum reprojection error of the marker based vSLAM is around 0.01 meters and it is being continuously calculated before the EKF, there may be a small unavoidable bias introduced even after calibration. This is one of the drawbacks of the loosely coupled system.

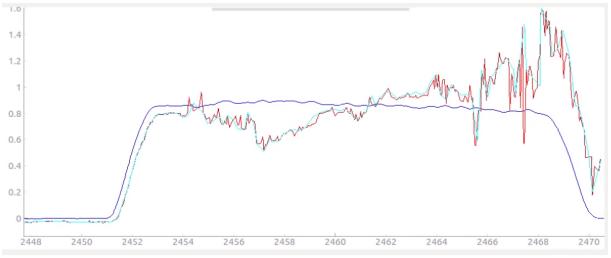
5.4 Experimental Results of Fused estimate

This section will describe some of the tests that were performed to evaluate the tracking performance of the loosely coupled EKF. The experimental setup is the same as the one for the SLAM setup with markers placed as shown in figure 4.3. The first test is

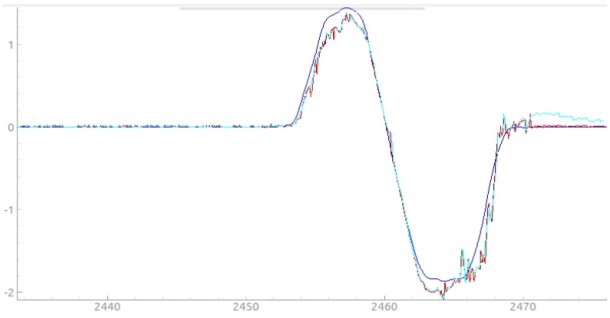
position tracking of the drone while performing the 'XYZ' maneuver similar to previous sections. The real time results are shown in figure 5.2. The fused tracking is very similar in performance to the SLAM estimate, but as seen in figure 5.2 (a) and (c) , the fused tracking (green) as implemented is not able to deal with the scale problem encountered by the SLAM estimate (red). With the same settings and weights in the process covariance matrix, the tracking was slightly worse compared to the SLAM estimate in certain scenarios as seen in figure 5.2 (c), where the fused estimate overshoots at the end. This is due to the fact that the IMU value is able to affect the estimate to cause it to be less noisy, but slow to converge to track the SLAM pose estimate, as the IMU odometry is less accurate. This problem can be solved by having a variable Q and R matrix depending on the motion of the drone and a high frequency IMU that is able to provide state information at a high resolution and low latency.



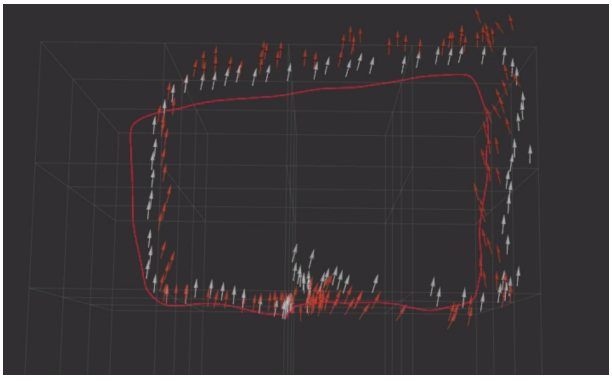
(a) x-position (meters) over time (seconds)



(b) z-position



(c) y-position



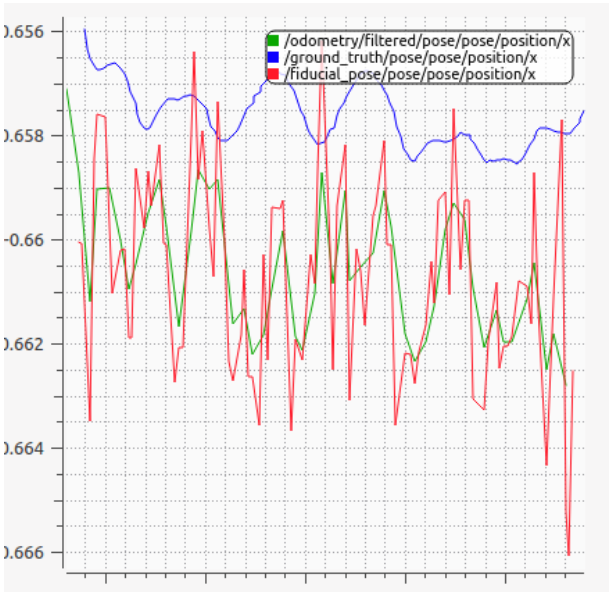
(d)

Figure 5.2: Fused estimate (green) of drone position in XYZ test compared to ground truth (blue) and SLAM estimate(red). Red arrows represent the SLAM estimate and white arrows the fused estimate in the square path visualization.

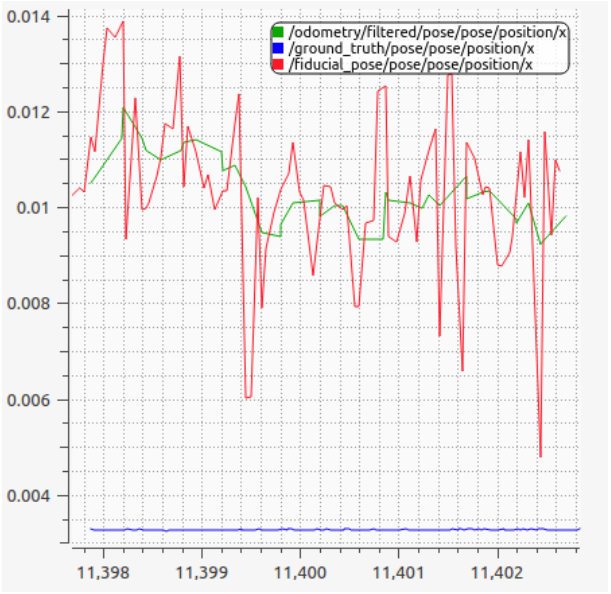
The second set of images in figure 5.3 show the results of the fusion test while performing small motions, similar to control corrections that the drone is expected to perform while trying to track a single point in the game environment as seen in Figure 5.3(a). The noise in the SLAM pose estimate (red) is high and is not able to track the small changes as seen in the ground truth (blue). Although the fused estimate (green) is not able to capture the motion in high resolution, it is still able to reduce the noise from the SLAM estimate. Figure 5.3(b) shows the result when there is no motion.

Some level of noise reduction is clearly visible in the fused x position. The process co variance Q can be reduced further to improve noise rejection, but the estimate will not converge fast enough match the new measurements. Figure 5.3(c) and figure 5.3(d) show the results of rolling and pitching with the fused estimate. The gyroscope readings measuring the rotation rate is incorporated in to the EKF, but due to the large delay and rubber banding effect, the scale is either overestimated or underestimated even though the overall rotation is captured.

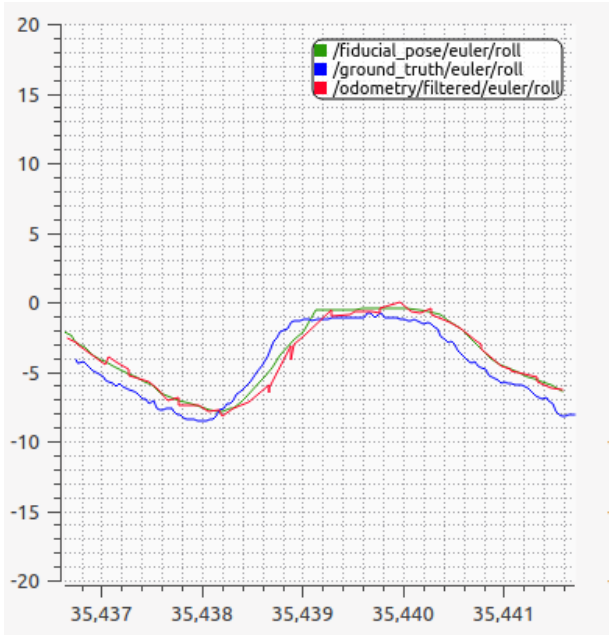
These results of the loosely coupled EKF sensor fusion overall displays some success, but it ultimately might not be enough in its current state directly incorporate into the mixed reality framework. The overall conclusions and possible steps towards this goal is detailed in the final section.



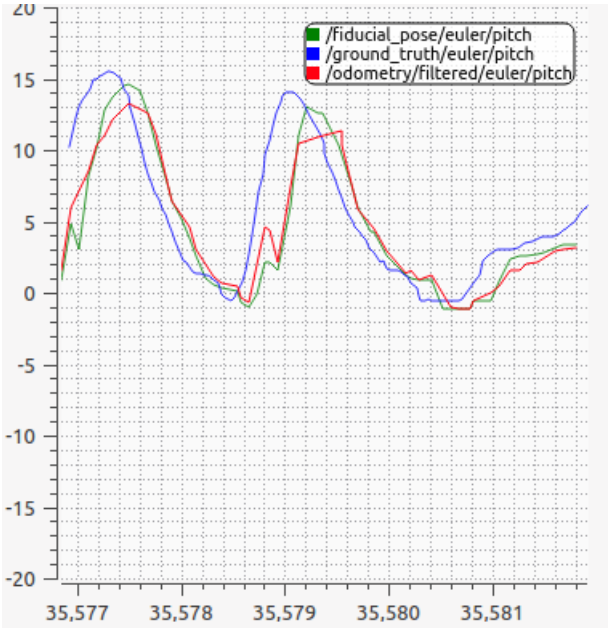
(a) Fused x position (fast motion handling)



(b) Fused x position (steady state)



(c) Roll tracking



(d) Pitch tracking

Figure 5.3: Fused estimate (green) of drone position in XYZ test compared to ground truth (blue) and SLAM estimate(red) in various tests

Chapter 6

Conclusions

In this thesis, a hardware and software framework for implementing a multiplayer mixed reality game using a quadrotor aerial vehicle was presented. The problem of localization of the aerial vehicle in an indoor GPS-denied environment is explored and an algorithm for Simultaneous Localization and Mapping using the camera sensor and bit coded fiducial markers is tested. Experimental results showed that the camera sensor alone is inadequate to account for sudden movements of the aerial vehicle and that the noise is generated when landmarks for the visual sensor are close to the limits of its range. Furthermore, the camera sensor does perform very well during changes in rotation, particularly when yawing. Another limitation was that the SLAM pose estimate could only register at less than 20 Hz after processing the camera sensor data.

In order to overcome these limitations, a loosely coupled EKF based sensor-fusion method that utilizes the data from the on-board IMU sensor was proposed and implemented. This system was tested in the lab environment and the pose estimate is compared with the ground truth and estimate from the camera-only readings in different movement scenarios as anticipated in the proposed mixed reality game. The sensor fusion method was able to perform better within a small time step scenario and have a similar to slightly worse performance in large scale tracking than vision algorithm separately. There is room for improvement in the performance of this filter by additional tuning and by improving some of the sensor broadcast capabilities.

6.1 Future Work

This work presented many challenges due to the process of starting the project from the point of a new business idea combining various new technologies with real constraints in hardware and software. The core work of building a framework for a drone-based mixed reality system and implementing an EKF based sensor fusion framework for improved localization is a good stepping stone in the road map for this ambitious project. The current monocular vision system limits both the player's natural depth perception and the localization accuracy and switching to a stereo camera can improve results dramatically. The IMU sensor on the drone is limited by the processing power of the drone and is only able to send measurements at a maximum rate of 10Hz. An improvement in the broadcasting rate would improve the performance of sensor fusion greatly. Adding additional sensors such as a depth sensor or an optical flow sensor measuring relative velocity to the sensor fusion algorithm will also improve the final pose estimate.

Other software level improvements could be to implement a tightly coupled EKF fusion system, where the visual SLAM and Kalman filter are implemented in a single algorithmic block. For better performance and avoiding latency issues, on-board implementation of the EKF with a dynamic system model should be considered at the cost of more expensive hardware.

References

[1] Pokemon Go. [Online]. Available: http://origin.pokemongo.com/

[2] Snapchat. [Online]. Available: https://whatis.snapchat.com/

[3] Mobidev. Augmented reality in healthcare: Way to medicine's digital transformation. [Online]. Available: https://mobidev.biz/blog/augmented-reality-in-healthcare-digital-transformation

[4] HTC Vive-VR. [Online]. Available: https://www.vive.com/eu/

[5] Playstation. PS-VR. [Online]. Available: https://www.playstation.com/en-us/explore/playstation-vr/

[6] Microsoft HoloLens. [Online]. Available: https://www.microsoft.com/en-us/hololens/why-hololens

[7] Oculus Rift. [Online]. Available: https://www.oculus.com/rift/

[8] E. Jonietz. Augmented Reality. [Online]. Available: http://www2.technologyreview.com/news/407473/tr10-augmented-reality/

[9] H. Ling, "Augmented reality in reality," *IEEE MultiMedia*, vol. 24, no. 3, pp. 10–15, 2017.

[10] F. Milgram, Paul Kishino, "A Taxonomy of Mixed Reality Visual Displays. IEICE Trans. Information Systems. vol. E77-D, no. 12. 1321-1329," 1994.

[11] Intel. Virtual-reality-vs-Augmented-reality. [Online]. Available: https://www.intel.com/content/www/us/en/tech-tips-and-tricks/virtual-reality-vs-augmented-reality.html

[12] M. M. Liszio S., "Designing Shared Virtual Reality Gaming Experiences in Local Multi-platform Games," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[13] R. Van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations," *International Journal of Virtual Reality (ISSN 1081-1451)*, vol. 9, p. 1, 06 2010.

[14] World Drone Prix, Dubai. [Online]. Available: https://youtu.be/gc3_wEB9wnI

[15] SIGGRAPH. [Online]. Available: https://www.siggraph.org/

[16] 3D Robotics. 3DR Solo. [Online]. Available: https://3dr.com/solo-drone/

[17] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer, "Exhaustive linearization for robust camera pose and focal length estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2387–2400, Oct 2013.

[18] H. P. Gavin. The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems. [Online]. Available: http://people.duke.edu/~hpgavin/ce281/lm.pdf

[19] OpenCV. Detection of ArUco Markers. [Online]. Available: https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html

[20] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320314000235

[21] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[22] H. U. Takafumi Taketomi and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, 2017.

[23] M. N. S. O. Davison AJ, Reid ID, "MonoSLAM: real-time single camera SLAM," *IEEE Trans 29(6)*, 2007.

[24] M. D. Klein G, "Parallel tracking and mapping for small AR workspaces," *Proceedings of International Symposium on Mixed and Augmented Reality*, 2007.

[25] P. Corke, J. Lobo, and J. Dias, "An Introduction to Inertial and Visual Sensing," *I. J. Robotic Res.*, vol. 26, pp. 519–535, 06 2007.

[26] "Real-time metric state estimation for modular vision-inertial systems, author=Stephan Weiss and Roland Siegwart," *2011 IEEE International Conference on Robotics and Automation*, pp. 4531–4537, 2011.

[27] E. S. J. S. Soatto, "Visual-Inertial Navigation, Mapping and Localization: A Scalable Real-Time Causal Approach," *Intl. J. of Robotics Research*, 2010.

[28] Z. T. J. Tian, Y.; Jie, "Adaptive-frame-rate monocular vision and IMU fusion for robust indoor positioning," *2013 IEEE International Conference on Robotics and Automation*, pp. 2257–2262, 2013.

[29] A. Erdem, A.T.; Ercan, "Fusing inertial sensor data in an extended Kalman filter for 3D camera tracking," *IEEE Trans. Image Process. 2015*, pp. 24, 538–548, 2015.

[30] L. Y. W. J. G. Jing, C.; Wei, "Fusion of inertial and vision data for accurate tracking," *Proc. SPIE 2012*, pp. 8349, 83 491D, 2012.

[31] T. Moore. Robot localization ROS package. [Online]. Available: https://github.com/ayrton04/robot_localization