Delft University of Technology
Master's Thesis in Computer Science

A Framework for Cooperative 3D Mapping of Unstructured Environments

Puneeth Nekkundi Somashekar

TNO innovation for life

embedded software

TU Delft Delft University of Technology

A Framework for Cooperative 3D Mapping of Unstructured Environments

Thesis

submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Embedded Systems

by
Puneeth Nekkundi Somashekar

Department of Physics and Electronics
Expertise center Technical Sciences
TNO
Stieltjesweg 1, 2628 CK Delft, The Netherlands

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

03 August 2011

A Framework for Cooperative 3D Mapping of Unstructured Environments

Author: Puneeth Nekkundi Somashekar
Email: puneethns@gmail.com

11 August 2011

Abstract

Cooperative nature in robots is a much sought after feature. Weakness of individual entities in the system could be over come by cooperation, which also brings in reliability and speed. Application of Computer Vision in robotics has brought in many new path breaking techniques, especially into aerial robotics. It has also managed to upstage the use of traditional inertial sensing methods for control and stabilization. With more computation power being packed onboard the robotic platforms, it is now possible to run some of the state-of-the-art Computer Vision and Control algorithms on the platform itself. We present a hybrid solution involving a vision based markerless Simultaneous Localization and Mapping algorithm and fudicial markers in a framework to achieve cooperative 3D mapping of unstructured environments.

Graduation Committee

Chair:	Prof. Dr. Ir. Arjan J.C. Van Gemund, Faculty EEMCS, TU Delft
Committee Member:	Dr. Ir. Stefan Dulman, Faculty EEMCS, TU Delft
Committee Member:	Dr. Ir. Tamas Keviczky, Faculty DCSC, TU Delft
Committee Member:	Dr. Ir. Zoltan Papp, TNO

Preface

Almost two years ago, I was preparing myself to leave for the Netherlands. I now see that I was hardly prepared for the events that unfolded next. It has been a great learning experience personally and academically, especially the last few months. Without help from the following folks, time here would have been more challenging.

I would like to first thank my supervisor at Delft University, Stefan Dulman, whose constant encouragement and persuasion has led me through this project. Meetings with him would leave me brimming with confidence. Many thanks to my supervisor at TNO, Zoltan Papp for giving me this opportunity, his critical remarks on my work and for reviewing this thesis while vactioning. Both have been extremely patient and instrumental during the draft of this thesis. My thanks to Tamas Keviczky and Masoud Dorosti of DCSC for their valuable time and inputs.

A special thanks to Arvid Halma at TNO, for his keen interest in my project, for answering my queries regarding Computer Vision, for all the fruitful discussions and reveiwing my work. My thanks to Arjan Kodde, who helped me in many ways. Cheers to Julio, Robin, Derek, Jeroen, Peter and all the folks at Acoustic Department for making my experience at TNO a pleasant one.

My parents, without whom I would not have been here. Their kind words, affection and encouragement have always got me through hard times. My friends, for making me feel comfortable during the first year of stay in Delft. Finally, I would like to thank Sowmya, my girlfriend, for being there for me when I needed her the most.

Puneeth Nekkundi Somashekar
Delft, The Netherlands
17th August 2011

Contents

List of Acronyms

CRC	Cyclic Redundancy Check
DQ	Dual Quaternions
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
MAV	Micro Aerial Vehicle
OM	Orthonormal Matrices
PTAM	Parallel Tracking and Mapping
PTAMM	Parallel Tracking and Multiple Mapping
RANSAC	Random Sample Consensus
SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
UQ	Unit Quaternions
VTOL	Vertical Take Off and Landing

Chapter 1

Introduction

1.1 Cooperative Robotics

In practical application of robotics, such as rescue operations, detecting forest fires, inspection of structures etc., having multiple robots would be beneficial. Possible robot failures exist, either due to an internal issue (e.g., failure of a sensor) or influence of the environment. In critical missions (e.g, rescue operation) multiple robots can guarantee reliability of the system. Some tasks such as exploration can be too complex for a single robot to accomplish. Other tasks such as mapping of an environment can be accomplished faster with robots coordinating with each other. In many cases, robots have inherent weaknesses such as limited power onboard, effects of which can be mitigated by deploying multiple robots. Building and using multiple low-cost robots can be easier, cheaper and more flexible than having a single powerful robot to perform a task.

Approaches to cooperative robotics can be broadly categorized into: (1) homogeneous and heterogeneous, (2) swarm-type and intentional cooperation [48] and, (3) hybrid cooperation. Robots in a homogeneous type are similar whereas, the heterogeneous differ in type and abilities. An example for a heterogeneous robotic team would consist of aerial and ground-based robots. In case of swarm-type large number of robot are used for a task and typically unaware of each others' actions, whereas in intentional, robots are fewer and interact with each other to be efficient at a task. Hybrid cooperation can comprise of a number of combinations of the previous two categories.

With many benefits, cooperative robotics pose many challenges or we may call them research opportunities. An early survey [49] outlines numerous research areas within cooperative robotics. Figure 1.1 gives an overview of the

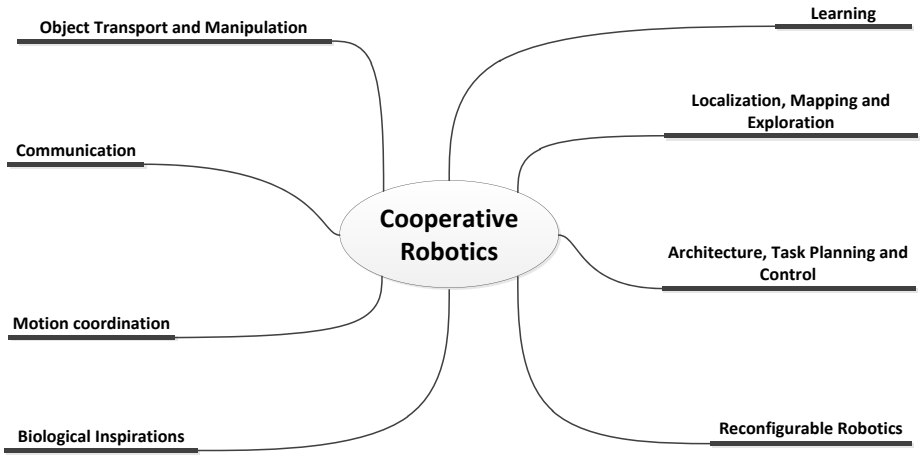


Figure 1.1: Research areas in Cooperative Robotics

survey. This research focuses on Localization, Mapping and Exploration area. Localization is a process of estimating the location of a robot with respect to a map, either a local or global map. Localization can also be relative i.e. between agents. Localization is an essential requirement for a task to be completed efficiently (e.g., mapping) and for intentional cooperation. Mapping involves gathering information from various sensors on a robot which is representative of the environment, either 2D or 3D. Localization and Mapping are interdependent. Simultaneous Localization and Mapping (SLAM) [18, 33, 43, 20] is a process where a robot incrementally builds a map of its environment and localizes itself in the map at the same time.

1.2 Visual Simultaneous Localization and Mapping

Camera has become an important sensor in all forms of robots, from hobby robots to large industrial ones. In many cases, it has become a primary sensor for a robot. The two autonomous Mars rovers *Spirit* and *Opportunity* use cameras for autonomously navigating and mapping the surface of Mars. The advantages of a camera when compared to other 3D rangers (e.g., LIDAR, Kinect sensor) is that its lighter, many times cheaper and has lower power requirement. These have been the prime motivating factors to use camera as the primary extroceptive sensor.

The Global Positioning System (GPS) is a satellite-based navigation system made up of a network of 24 satellites. GPS was originally intended for military applications, but later was made available for civilian use. GPS has been a

Figure 1.2: ARDrone Quadrotor

default aid for autonomous navigation of robots outdoors. In GPS denied environments (e.g., indoors), autonomous navigation of a robot is usually made possible with vision or 3D range information of the surroundings. There are several publications which address this scenario [4, 72], however, we focus on Visual Simultaneous Localization and Mapping.

Visual SLAM aims at recovering camera pose as well as map visual feature in the environment. A SLAM system which uses only a single camera is called monocular SLAM [33, 20, 14]. In this research we use Parallel Tracking and Mapping (PTAM) as the visual SLAM. Parallel Tracking and Mapping algorithm is a monocular SLAM. PTAM does not use any inertial sensing, hence, its also known as pure visual SLAM. It is a marker-less tracking system which also works in an unknown environment. Along with camera pose estimation, it also maps visual features in the environment.

1.3 Unmanned Aerial Vehicles

There has been a tremendous interest in autonomous Unmanned Aerial Vehicle (UAV) for both military and civilian applications [25, 71, 41]. Quadrotor helicopters are a class of vehicles under Vertical Take Off and Landing (VTOL) rotor-crafts category. It has two pairs of counter-rotating rotors with fixed-pitch blades at four corners of the airframe. Quadrotors have become indispensable in aerial robotics, typically have a span ranging from 15 cm to 60 cm. They are cheaper than their cousins, Micro Aerial Vehicle (MAV) which have a span less than 15cm and weigh less than 100g and have low risk of being seriously damaged (e.g., DelFly[15]).

Quadrotors are ideal mobile platforms in urban and indoor scenarios. They are small enough to navigate through corridors and can enter structures through windows or other openings and hence, make an excellent platform for surveillance, aerial inspection, tracking, low altitude aerial reconnaissance and other

applications.

Inspection of structures and searching for contamination in an environment are some interesting applications where quadrotors have an advantage over other robots. Quadrotors can easily navigate through a cluttered environment and they're capable of explore both indoors and outdoors.

Quadrotors come with their own set of limitations, namely, limited payload, flight time and computational resources. Quadrotors are inherently unstable and need active stabilization for a human operator to fly them. Quadrotors are generally stabilized using feedback from Inertial Measurement Unit (IMU). The active stabilization loop (low level control) can range from a simple implementation on microcontroller [42] to a dedicated hardware implementation on Field Programmable Gate Array (FPGA) [10].

Autonomous flight capability based on Computer Vision techniques (e.g., PTAM) require much more computational resource. Recently, more computation power is being packed onboard, e.g., Pixhawk Cheetah[50]. This platform boasts a dual core processor onboard, in all other respects it is a flying laptop. Considering vision based autonomous algorithms using major amount of computational power, we want to keep our computational use to a minimum while achieving real time performance and also allowing room for other applications running onboard to interact with the environment.

1.4 Problem Statement

Consider a scenario where autonomous quadrotors have a visual SLAM algorithm running to produce 3D map of point features in their own reference frame. These agents do not need to be in communication range, though some portions of their paths overlap.

The global map, which is a combination of map generated by the quadrotors is a representative of the environment to be mapped. For aligning these maps we must first establish a geometric relationship between them (a transformation). With the knowledge of transformation agents can share their 3D map, further cooperate to map unexplored regions in the environment and improve the global map. A consequence of aligning the maps of the quadrotors is that they can localize each other in their maps.

The research questions answered in this thesis are:

- *Can 3D mapping be achieved using a low cost quadrotor?*
  In literature, much of the work on 3D mapping using a quadrotor have made use of expensive and specialized hardware. We would like to know if 3D mapping can be achieved using an inexpensive consumer product.

- *How are sparse noisy 3D world models aligned?*
  Previous work has made use of laser scanners to collect 3D information

which is usually dense and very precise. This work uses visual SLAM which generates sparse noisy 3D data. We would like to know how such noisy sparse 3D data can be aligned in computationally efficient manner.

- *What are the possible system architectures for cooperative mapping?* Given that an environment can be mapped cooperatively with PTAM, what are the possible configurations of the system with quadrotor as a platform?

1.5 Thesis contribution

The contribution of this thesis is threefold:

- An important aspect of cooperative robotics is the use of multiple low cost robots to do a complex task. For this reason, a low cost quadrotor was chosen and the performance of the visual SLAM with this platform was evaluated as a part of this thesis.
- For agents in a cooperative robotic mapping team to contribute to a global map, it becomes necessary to bring all the agents into a common reference frame. To this end, we propose a real-time and computationally inexpensive method to establish geometric relationship between sparse world models produced by the visual SLAM using markers.
- We propose three architectures for cooperative 3D mapping based on visual SLAM and characterize these architectures at high level. These architectures address aspects such as communication, (computational) power and payload capability of the platform.

1.6 Thesis outline

Rest of the thesis is organized as follows: In chapter 2, we start with a discussion on work related to cooperative 3D map building. It is followed by background information on the platform used in this thesis, the monocular SLAM PTAM and a discussion on marker based tracking systems.

Chapter 3 discusses techniques that are used for aligning point clouds and we see how we can make use of markers as landmarks to align sparse point clouds. In chapter 4, we identify functional blocks of an agent in the system and configure them to form different architectures. It is then followed by an implementation of a scheme.

Chapter 5 we evaluate the performace of the marker tracking system and the monocular SLAM with the video stream from the onboard camera. We also align two sets of scenes that origin from different agents using the technique described in chapter 3. Finally, we end with some conclusions and future work.

Chapter 2

Related Work and Background

3D mapping of an indoor environment is a well researched topic and is addressed with a variety of solutions along with different combination of sensors [12, 78, 7]. Generating 3D maps of indoor in real-time can be very useful especially in a rescue scenario, where the rescue worker can get an initial overview of the situation. 3D mapping of an unknown environment by an autonomous robot can be time consuming and a complex task. Cooperation among the robots can potentially speed up and ease the task. Most research has been focused on generating maps using a single quadrotor [19, 4, 77]. There are two key issues to be addressed in the context of cooperative 3D mapping;(1) creating a locally consistent map on the each of the robots and (2) combining these locally built maps into a global map.

2.1 Related Work

We first investigate an early work on 3D cooperative mapping exclusively using laser range finders targeting indoors. The visual SLAM PTAM has been primarily used as localization system onboard a quadrotor [8, 1]. Weiss et al. [77] use the 3D feature map and keyframes generated by PTAM to create an intuitive map of the explored environment. However, they do not explore application of PTAM for cooperative mapping or localization. A Kinect sensor onboard a MAV is also an attractive option for dense 3D mapping of an indoor environment. We look at one such application.

Occupancy grid map is a classic representation of a map in which a robot assigns a x-y coordinate (in 2D) a binary occupancy value which indicates an

Figure 2.1: This picture show the 2D laser scanner and the 3D rotation mechanism on the robot [62].

object is present or not. Occupancy Grid Mapping[73] are a class of algorithms in probabilistic robotics for mobile robots which address the problem of generating maps from noisy and uncertain sensor measurement data, given that the robot pose is known. In occupancy grid mapping, a map represents a field of random variables, arranged in an evenly spaced grid. Each random variable is binary and corresponds to the occupancy of the location it covers. Occupancy grid mapping algorithms implement approximate posterior estimation for those random variables. It is a popular method to build maps using robots [63, 73, 78, 40]

Ryde et al. [62] use multiple land based robots which cooperate to localize each other and build a 3D global map of an environment. Ryde et al. uses agents with a custom laser ranger setup and an algorithm that gives 3D data (volumetric picture element or *voxels*) of their environment to build 3D occupancy grid. The robotic team mapping the environment consists of two types of robots: mapping robots and stationary robots. The mapping robots Figure 2.1 localizes itself in the environment using retroreflective (a surface which reflects light with minimum scattering) cylinders used as beacons in Figure 2.2 and stationary robots. A stationary robot can be identified by the circular feature introduced by its retroreflective cylinders in the range data. By locating this feature in range data, an exact position of the stationary robot can be extracted.

Further, these stationary robots are uniquely identified as landmarks. The challenge is to align successive laser scans and build a 3D map of the environment on the robot. This is done by locating two landmarks in a scan, the same landmarks are identified in the subsequent scan. The change in pose is found that is, its rotation and translation is found between the scan and added to the global map. The robot can also use this information to localize itself with

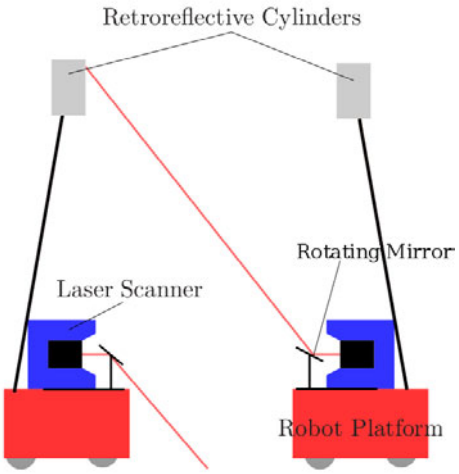


Figure 2.2: This figure show the retroreflective cylinders attached to the robots. A stationary robot on the left can be located by robot on the right by looking for a circular feature introduced by retroreflective cylinder.

respect to landmarks. This work has been extented further by incorporating mutual observations of the robots.

An important contribution of Ryde el al. is use of agents as landmarks and to do scan matching. This eliminates the need for successive scan matching to create a global map, which is an expensive operation in terms of computation power and memory. Using agents as landmarks can be useful in mapping an unknown environment. Scan matching using agents as landmarks is only suitable for land based robot as they can remain stationary for long periods of time. The laser range finder and rotation mechanism would be too bulky for a quadrotor.

Laser scanners can produce highly accurate geometric models of the environment. Light weight 2D laser range finder[36] are available which can be easily mounted on the quadrotors. Laser range finders onboard quadrotors are being predominantly used for localization in 2D as well as mapping (2D SLAM), obstacle avoidance and path planning[5, 27]. 3D map of an environment can be reconstructed by layering the 2D scan [4], however, these units are expensive, consume a lot of power, and thus have a huge impact on the flight time of the quadrotor.

Little et al. [40] present a system in which two robots sharing a world coordinate frame build a 2D occupancy grid map. 3D information of the environment is obtained using a calibrated stereo camera setup. Camera setup provides a disparity image, difference of the two view of stereo camera. Brightness in the

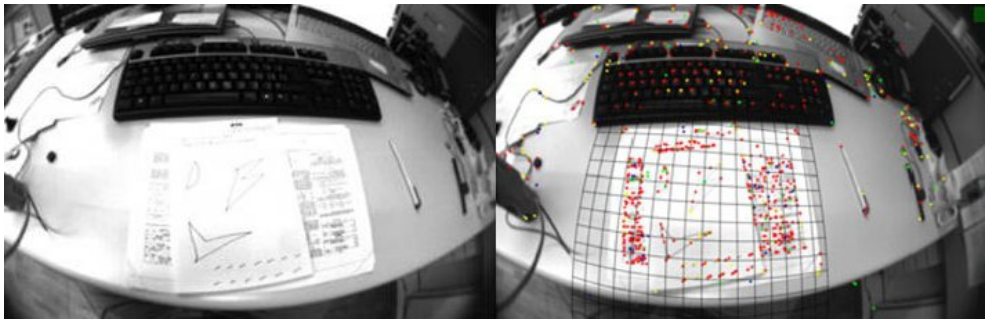


Figure 2.3: The picture depicts a sample scene on the left. On the right is the view of the scene through PTAM after being mapped. The color dots represent the features in the scene which have been added to the map. [77]

disparity image indicate distance to object from camera. This image is then compressed into 1D map consisting of maximum disparity versus image column called a radial map. With information from the radial map, occupancy grids are updated. The edges found in the 2D occupancy grid called "corners" are used for localization of the robot in the map. A specific pair of corners is used as *home base*. The robots starts with initializing itself with respect to a home base, hence, share the same coordinate frame. They exhibit a simple cooperation by contributing their updates to a shared map.

Authors of [57] use stereo camera measurements on land-based robot to build 3D occupancy grids of the environment and participating robots share this information to efficiently explore an environment. However, they assume robots are externally localized through a global localization system.

Weiss et al. [77] use the visual SLAM PTAM to build a 3D mesh of the environment using point features, which is then textured to give an intuitive feel of the environment to the user. Weiss et al. also use PTAM with a Linear Quadratic Gaussian Controller with Loop Transfer Recovery (LQR/LTR) [8] for autonomous (trajectory following) vision based navigation onboard. Our work can be considered as an extension to the autonomous setup used in this work, essentially adding more agents and enabling cooperation among them for mapping.

Weiss et al. use the sparse 3D point cloud from PTAM to create a 3D mesh. The mesh generated is proposed to be used for autonomous obstacle avoidance. This can be accomplished by preventing the quadrotor from flying through the mesh. In Figure 2.3 a scene is mapped by PTAM, the color pixels represent features which have been added to the map. First step in the process is to project the 3D points generated by PTAM on to a main plane [77] to reduce the dimensionality. Reducing dimensionality brings down the computational time. Delaunay Triangulation [67] is run on these points to generate a 2D mesh. The third dimension is then added using only the edge information

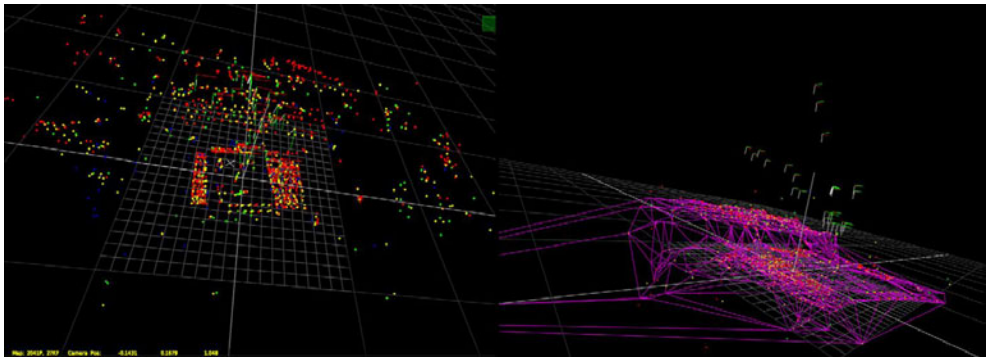


Figure 2.4: The point cloud on the left is the result of the scene in Figure 2.3 being mapped by PTAM. On the right, the point cloud is then meshed. Notice the elevation in the mesh, this is due to the keyboard in the scene. [77]

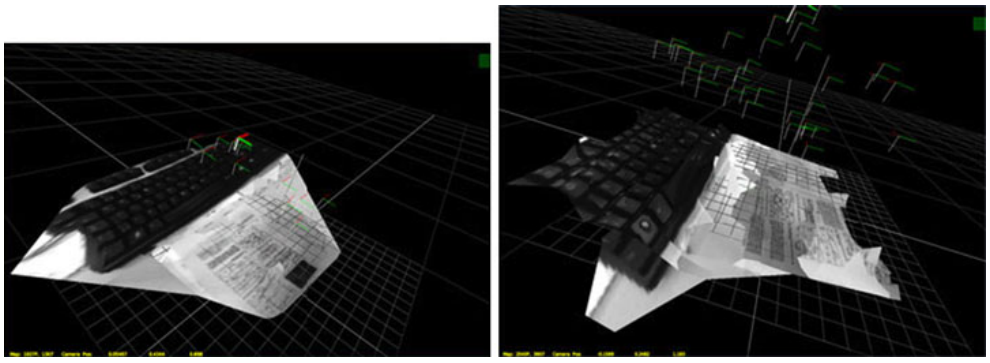
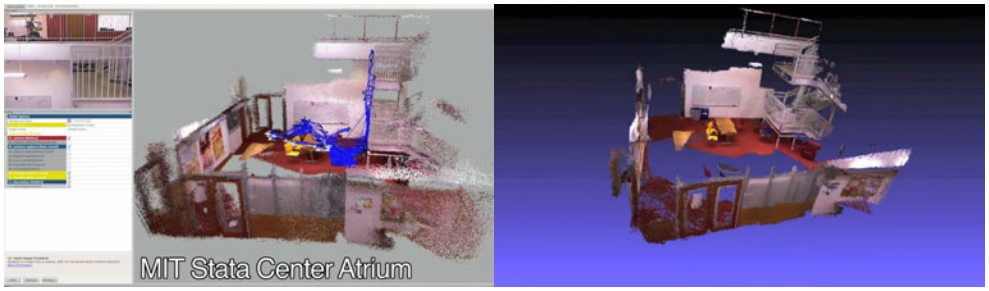


Figure 2.5: The 3D mesh is then textured with keyframes from PTAM, on the left with low and on the right with high resolution image. [77]

of Delaunay Triangulation. Once the 3D mesh is ready, the keyframes from PTAM are then used to texture the surface of the mesh (Shown in Figure 2.5). Mesh generation and texturing is done off-board on a 2GHz dual-core laptop in real-time, which also runs the visual SLAM.

Kinect sensor[65] is an inexpensive sensor from recent times that is capable of providing accurate 3D range information using a continuously-projected infrared structured light (a known pattern of pixels). Figure [2.6a] shows a quadrotor with kinect onboard developed by the Robust Robotics Group, MIT[56]. Visual odometry is a process of deducing camera pose relative to incoming images, this is done onboard in real time and is used along with IMU to control the quadrotor. This enables fully autonomous navigation in GPS denied environment. Kinect in addition to providing depth information for every pixel of camera frame also provides primary color values per pixel (Red, Green and Blue). The combination of primary colors and depth map is referred

(a)



(b) (c)

Figure 2.6: (a) shows the quadrotor with Kinect onboard. (b) and (c) 3D model of a scene created using the RGBD information form kinect.

to as RGBD. A SLAM which uses this is referred to as RGBD-SLAM. This involves extracting features common to two pairs of successive RGB images, obtain 3D transformation using Random Sample Consensus (RANSAC) and align the depth information. The RGBD-SLAM in this setup is done off-board meanwhile relaying corrections in position to quadrotor and maintain global consistence. Figure 2.6b and 2.6c show the results of RGBD-SLAM. The results of this implementation are yet to be published.

Discussion

Most of the cooperative mapping schemes have been aimed at 2D environments [16, 40, 53]. Cooperative indoor 3D map building has been demonstrated only on land-based robots using laser scanners [62, 72] and also with a stereo camera setup [57]. We can conclude that not much work has been done with respect to cooperative indoor 3D mapping.

As memory usage increases with the increase in resolution of the grids, memory cost becomes a general concern with occupancy grids. Computational and memory costs increase beyond the limits of real-time performance often in case of 3D. Exchanging large occupancy maps among agents can also pose problems.

For a platform like a quadrotor with payload and power restrictions, laser scanners are generally not an option. A stereo camera setup has been used for visual odometry in unstructured environment [4], however, it does address the issue of creating 3D maps with stereo cameras. The stereo camera rig can also pose problems for a quadrotor with payload restrictions. Also, authors of [4] use laser scanner for creating 3D map of the environment, but do not explore cooperative 3D mapping.

A camera is an ideal sensor for a quadrotor as it is light and requires little power. In recent times, significant progress has been made by the computer vision community with respect to dense 3D reconstruction in real-time using a monocular camera. Authors of [70, 44] have shown that a dense depth map of a scene can be produced using only a monocular camera. However, it involves huge computational costs.

A monocular SLAM like PTAM can generate sparse 3D map of features in the environment in real time with modest computational requirement. Exchanging map information is also easy as map data only consists of a point cloud and is sparse in nature. Along with a monocular SLAM, the combination of camera and quadrotor is inexpensive in generating indoor 3D maps. With a cooperative effort we can map an environment in greater detail.

Figure 2.7: Parrot ARDrone

2.2 Background

ARDrone

The ARDrone quadrotor (called a drone) from Parrot[2] is a consumer grade product which is low cost and easy to use. It comes with an "indoor hull", which covers the propellers and can therefore be safely used indoors. All parts of this quadrotor are replaceable including the onboard computer. It is well built and can survive some serious crashes.

The onboard computer is Wi-Fi enabled, which makes it easy to control the quadrotor with any Wi-Fi enabled devices such as smart phones, tablets and PCs. The firmware and hardware onboard are closed. However, it comes with a Software Development Kit (SDK) which gives easy access to sensor data and control software onboard. The software development kit has been continuously evolving since its initial release. There is an active community[51] of users and developers.

ARDrone SDK

ARDrone being a consumer product is aimed at augmented reality games. The ARDrone SDK provides necessary tools and resources to create an augmented reality or a control application. Targets include Linux, Windows and iOS. The SDK is written in *C*, but designed using object oriented paradigm, which makes it difficult to modify low level features of the SDK. Figure 2.8 gives an overview of integrating custom code into the ARDrone SDK 1.5.

ARDrone SDK takes over the task of setting up a connection to the drone once the user joins the ad-hoc Wi-Fi network of the drone. The function *ardrone_tool_setup_com()* sets up a *TCP* connection to the drone from the device where the application is running. This connection is used for for sending critical data to the drone and receive heartbeat signal from it. *UDP* connections are used for sending attitude commands, receiving video frames and sensor data. *ardrone_tool_init()* initializes the internal objects which pack and unpack data to and from the drone. At the core of the application, three threads handle user input, telemetry data and video frames receive events. For more details refer the developers guide [17].

Hardware specifications

Actuators:	Four brushless motors (35,000 rpm,15W)
Embedded Computer:	ARM9 468 MHz DDR 128MB 200MHz Wi-Fi b/g enabled
Operating System onboard:	BusyBox v1.14.0 Linux v2.6.27.47
Onboard sensors:	3 axis accelerometer 2 axis gyrometer 1 axis yaw precision gyrometer Ultrasound Altimeter
Forward looking camera:	93 degree wide-angle diagonal lens
Downward looking camera:	64 degree diagonal lens
Battery:	Lithium polymer battery (3 cells, 11,1V, 1000 mAh)

The onboard computer uses the inertial sensors (accelerometer and gyrometers) for stabilization (low level control) along with downward looking camera which provides velocity estimates of the quadrotor in XY plane of the quadrotor body-fixed frame using optical flow. The flight time of the quadrotor is poor (only 9 min). The drone may drift a lot on some surfaces in which case

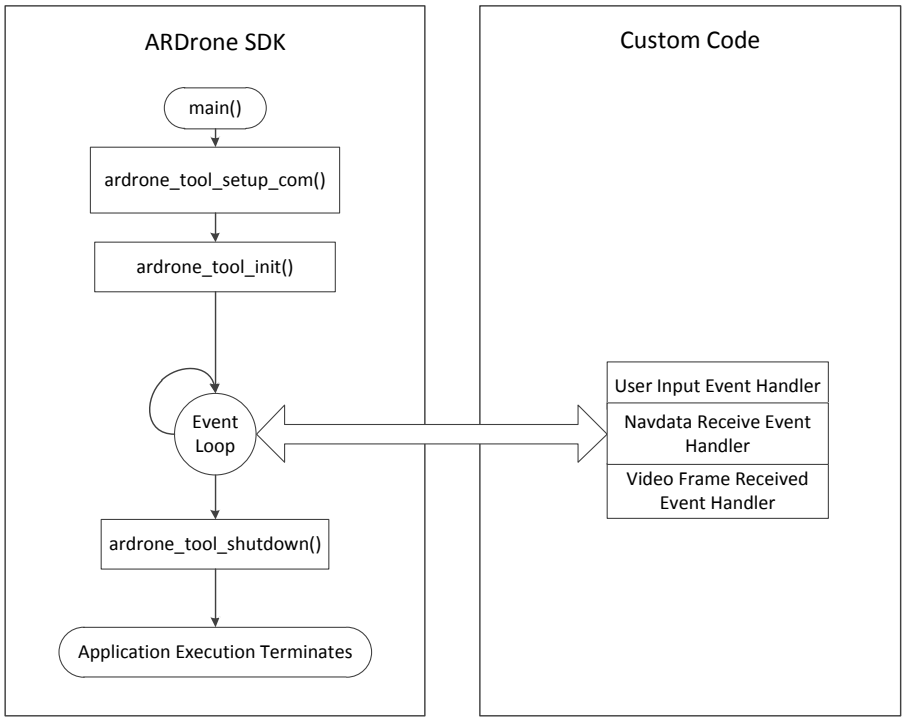


Figure 2.8: Custom Code Integration into ARDrone SDK

placing some newspaper with features may help.

ARDrone being an inexpensive product is at the low end of the quadrotor market spectrum. Mid-range quadrotors such as Mikrokopter [42] offer much more flexibility in hardware and performance. High end of the market includes platform such as [71, 25, 41] which are widely used in many quadrotor research laboratories (e.g., [35, 66]) around the world. ARDrone quadrotors are easy to work with and does not require much effort during initial setup.

Parallel Tracking and Mapping (PTAM)

There have been many projects around monocular SLAM, some of the interesting ones are MonoSLAM[14] the very first one; Eade et al.[20] and Parallel Tracking and Mapping[33]. The MonoSLAM is based on Extended Kalman Filter (EKF) SLAM[69], while Eade el al. based their's on FastSLAM2.0[43].

The main feature of PTAM algorithm [33] is that the simultaneous localization and mapping has been split into two parallel threads, Tracking and Mapping. Most modern processors have at least two cores, hence, it is possible to run each of these threads on a core.

Tracking thread tracks features in the incoming frames and estimates the pose the camera. A select few frames called *keyframes* are passed over to mapping thread, which triangulates point features in these images and builds a 3D map. Separation of tracking and mapping enables each of the threads to run at a different rate. The tracking thread runs at the rate of incoming frames (upto 30 Hz). While the slow mapping thread runs at a much slower rate, since much of the information between two successive frames is redundant. The map generated by PTAM consists of a collection of point features located in a world coordinate frame $\mathcal{W}$. Each point feature represents a locally planar textured patch (8x8 pixels) in the world. The jth point in the map ($\boldsymbol{p}_j$) has coordinates ($x_{j\mathcal{W}}$ $y_{j\mathcal{W}}$ $z_{j\mathcal{W}}$), a collection of such points is commonly referred to as point cloud.

Computer vision terminologies

Before we look into the working of the PTAM algorithm, we need to understand a few computer vision terminologies. For more detailed description refer to [28].

Camera calibration is a process of finding the true parameters of the camera that produced a given image which includes finding focal length of the lens, true camera image center and others. Camera calibration is a necessary step to extract 3D information from 2D images.

Essential matrix relates to corresponding points in a stereo pair (images of a scene from two different views) and is used to determine both the relative position and orientation between the camera views and the 3D position of corresponding image points, given that the camera is calibrated.

Image Pyramid is a multi-scale representation of an image in which image is subjected to repeated sub-sampling in resolution to create different levels with an application of filter if required. If the full resolution of the frame is 640x480 then this is called level 0. The original frame is further sub-sampled to 320x240, 160x120 and 80x60 for a four level representation.

RANSAC is an iterative method to estimate the parameters of a certain model starting from a set of data contaminated by large amounts of outliers. It was first conceived to solve Location Determination Problem (LDP) [24], where we determine that point in space from which an image was obtained, given that we know the locations of a set of landmarks.

Triangulation is a method to determine a point in 3D space given that we know the projections of the point on two, or more, images.

Initialization of PTAM

The camera was first calibrated using an application which accompanies the source code of PTAM. For the tracking thread to start a map has to be first initialized, essentially to establish an origin in the environment. The initial

map is build using a stereo pair. User must first hold the camera against a planar textured surface, on pressing a key PTAM picks up the most prominent 2D image patches in the current frame (first keyframe) and tracks them in the subsequent frames. When user presses a key again after translating the camera while making sure the 2D image patches are not out of the frame, this frame is recorded as second keyframe. The two keyframes provides two views of the same features. By applying five-point algorithm and RANSAC [24] on these corresponding features an essential matrix is obtain. These features are then triangulated and a base map is created. A plane is fit through these points using RANSAC, which is called the dominant plane. This map has the camera at the center of the origin. It is then rotated and translated to have the dominant plane at $Z = 0$. This plane does not play any special role in the system, but only used in augmented reality application.

Figure 2.9 shows how PTAM is initialized. The grey wireframe seen in Figure 2.9b represents the dominant plane.

Tracking

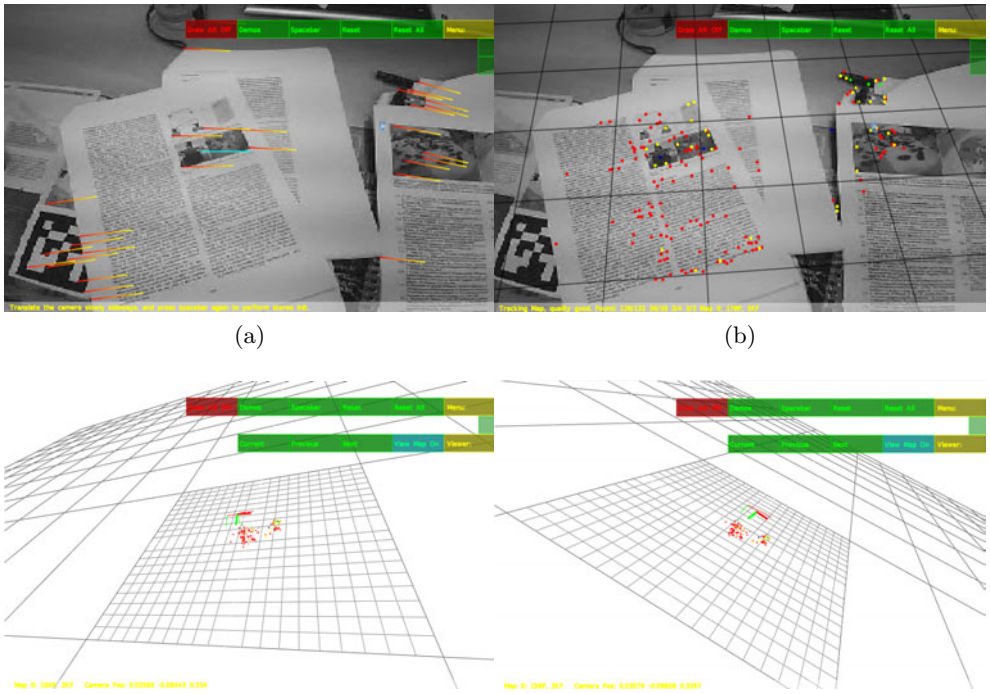
PTAM tracks the pose of the camera with respect to the reference frame as shown in Figure 2.9c and 2.9d. Tracker (tracking thread) can estimate the pose of the camera by tracking feature points (mappoints) that are present in the map to those in current frame. Features are detected at two scales: coarse and fine scale. Based on a motion model (decaying constant velocity [33]) tracker first projects possible locations of small set of mappoints which are present in the stored map (a coarse estimate) on to the current frame. Using the coarse estimate camera pose is updated. A larger set of features are then projected on to the frame and searched. With this information pose is further refined and updated. Using a motion model to estimate possible location of features in a frame is easier and faster than doing an exhustive search.

For tracking, the incoming RGB images are converted to 8 bits per pixel grayscale images. An image is reconstructed into a four levels image pyramid. PTAM uses Features from Accelerated Segment Test (FAST) [59] corners as interest points. A patch of 8×8 around these corners is used for feature matching.

Mapping

After map initialization, new keyframes and new feature points are added to the map as the camera moves over unexplored space. New keyframes are inserted into the map based on these heuristics [33]:

- Tracking quality should be good i.e. at least 50% of the features are being tracked.
- It has to be more than twenty frames since last keyframe was inserted.



(a) (b)

(c) (d)

Figure 2.9: Initialization of PTAM: (a) 2D image patches being tracked during translation. (b) Mappoints are initialized and base map is created. (c and d) These figures shows the view of base map. The colored lines in the picture represents the current position of the camera with respect to the world coordinate frame. The colored dots represent the feature points.

- The camera must be at a minimum distance away from a nearest keypoint already in the frame.

Tracking thread would have passed over a keyframe (when above conditions are met) with pose estimate. However, not all observed features would have been used to update the pose in tracking thread, so mapping thread re-projects the feature points, calculates pose and includes additional measurements. FAST corners calculated by tracking thread are further narrowed down to a set of salient feature points based on Shi-Tomasi[68] score. Only new feature points are selected as map points while discarding existing ones. Depth information of the new map points has to be estimated, which is done by triangulation with another keyframe already in the map.

PTAM uses bundle adjustment technique to refine the map, a batch optimization which is computationally expensive but highly accurate. Bundle adjustment essentially minimizes the re-projection error of mappoints and

the corresponding observations in keyframes by minimizing a robust cost function[33] based on image errors. Bundle adjustment is done at two levels global and local. A global bundle adjustment is done for all the keyframes and all the map points in the map, which is time consuming. Global bundle adjustment is handy when camera is not exploring but it becomes rather restricting when new keyframes are added in quick succession, which is often the case. For this reason, local bundle adjustment is included which applies the technique to a subset of recent keyframes and associated map points. The map is also further improved using data association refinement. This is only done when bundle adjustment has converged and camera is in a well explored region. Here new measurements are made in old keyframes and outliers are reassessed.

Another important component of this tracking system is the relocalizer. When tracking fails, relocalizer attempts to localize the camera in the map. This relocalizer implementation exploits the relatively dense distribution of keyframes. A descriptor for each keyframe which is a four level image pyramid of the keyframe with Gaussian blur applied to each level is created. When tracking fails relocalizer compares keyframe descriptors against the current camera image. If a match is found, tracker is reset to that position in the map. During experimentation pf PTAM with video stream from the drone, it was noticed that when the keyframe distribution was minimum, relocalization was not very successful and had reset the tracker to a wrong location in the map and tracking continued from wrongly initialized position.

The important feature of PTAM is that it works even in an unknown environment. PTAM takes advantage of large number of feature point to create accurate maps using local and global batch optimization. PTAM does not deal with uncertainties be it features, map, or pose, which becomes an overhead in most cases. It is also robust against partial camera occlusion (at least 50% of features should be visible). It is a pure visual SLAM without the need for any inertial sensing. Both EKF and particle filter [52] scale poorly with number of feature points in the map. They can only handle a few dozen points in the map. [33] estimate PTAM's practical limit is around 6000 feature points and 150 keyframes. This project is open source, which makes it attractive and has an active community of users involved in it.

Marker-based Tracking

Optical tracking of an object in a scene or tracking a camera in a scene is a widely researched area. Optical tracking of an object can be 2D (within the camera frame) or 3D pose (with respect to a reference frame). In augmented reality, an essential task is to estimate 3D pose of the camera with respect to its scene, in order to project virtual objects in the scene. Similarly for precise control of a autonomous robot, location information is vital. If the camera and robot's reference frames coincide (or relationship between them are know), we

Figure 2.10: A Matrix Marker

Figure 2.11: A CyberCode Marker

can apply solutions from this area to solve localization problem.

Fiducial or fiduciary marker is an object placed in a scene, used as point of reference or measurement for an imaging system. Fiducial image or more commonly referred to as Fiducial marker-based tracking systems are popular among augmented reality enthusiast [54, 32, 55, 75, 23]. Fiducial marker-based tracking systems are not just limited to the augmented reality circle, it has made its way into aerial robotics[60, 50] as well. Experimental results from Lee et al. [37] have shown that marker-based solution for visual pose estimation of quadrotor is quite reliable. Some of the passive fiducial marker-based tracking systems are discussed next.

Rekimoto [54] was the first to develop a Binary encoded marker called Matrix Marker. The rationale behind using binary encoded information in the marker was to have large number of possibilities (2^{16} combinations are possible) and accurately identify each of them, which included an error check. Design of the marker involves a square black border (called the 2D-code frame) and a combination of white and black squares in the interior. These square are encoded with the binary data used to identify the marker. On detection, the binary data is decode by sampling the code area of the marker. ID of the marker is obtained after performing error check on decoded data.

A successor to Matrix from Rekimoto was dubbed CyberCode [55]. The basic design of the marker was changed. 2D-code frame was removed and a guide bar was used as an aid for detection. Four black squares on the corners of 2D-code frame were left to recover pose while blocks around these were left with white squares, rest of it being part of the code. It is possible to concatenate two markers (in order to encode more information on the same) by having binary data on two sides of the guide bar.

On similar lines to Matrix, ARToolKit a software library for marker tracking was created by Kato et al., in a framework for augmented video conferencing[32]. Marker consisted of a black frame and outer corners of the frame were used for pose estimation of the camera similar to Matrix. Instead of binary data in the code area, any arbitrary pattern can be used. The marker is identified using template matching. Once the pose of the marker is estimated, the interior of the marker is compared against a set of templates in memory. This

Tracking System	Robustness	Identification	Source Code
Matrix	-	Binary Encoding	Not Available
CyberCode	-	Binary Encoding	Not Available
ARToolKit	-	Template Matching	Open
ARTag	++	Binary Encoding	Closed
ARToolKitPlus	+	Binary Encoding	Open

Table 2.1: Comparison of Fiducial maker Trackers

immediately becomes a problem, since loading large number of markers needs more memory and comparisons, while increasing inter-marker confusion. The software library is open sourced and is very popular.

ARTag[23] markers developed by Fiala used digital encoding theory to make identification more accurate and reduce inter marker confusion. Among all the marker system discussed here, only ARTag is capable of handling partial occlusion of the marker edges. Grayscale thresholding technique for detecting markers used in ARToolKit would fail when a marker is not uniformly illuminated. ARTag uses edge detection; edge pixels are thresholded and linked into segments. This potentially mitigates the effects of non uniform illumination of markers. Forward Error Correction (FEC) and Cyclic Redundancy Check (CRC) is performed on the data decoded from the marker. ARTag provide 2002 makers, this set was chosen to maximize Hamming distance (number of bits that differ in two binary code) between them.

ARToolKitPlus[39] is an improved version of ARToolKit, developed specifically for mobile devices[75] like Smart phones, PDAs etc. ARToolKitPlus essential wraps ARToolKit pose tracking library into a Class-based API (in C++). Owing to its object oriented nature, ARToolKitPlus is capable of tracking multiple markers, while not affecting tracking performance. ARToolKitPlus uses marker inspired by ARTag[23]. A robust pose estimation algorithm[64] has been used to improve pose estimation and reduce jitters. It also features automatic thresholding for marker detection, which mitigates the effects of non-uniform illumination of marker. In addition to template based matching similar to ARToolKit, it also has simple id markers (512 combinations) and BCH [9] coded markers with built-in FEC (4096 combinations).

From table 2.1 it is clear that ARTag is more robust in terms of detection, identification and can handle partial occlusion, however, it is a closed project. Though ARToolKitPlus does not offer the same level of robustness, it offers good performance (in terms of memory and computational speed) and being open sourced[46] makes it our choice.

Chapter 3

Aligning Point Clouds

Aligning sets of 3D data is an essential and basic problem being tackled in Computer Vision as 3D-3D Registration Problem. Most common method used to tackle this problem is Iterative Closest Point (ICP) algorithm[6]. Primarily ICP is used for 2D/3D reconstruction from multiple scans of 2D/3D scanners. It is often used in robot localization. There are numerous variation[61] of the algorithm addressing a variety of scenarios, however, there are some inherent shortcomings which we discuss in the coming section. First some theoretical concepts are presented.

3.1 Rigid body transformations

A rigid transformation or Euclidean transformation is function from and to Euclidean space which preserves distances between the points[26]. A point cloud is an example of rigid body, wherein distances between the points are preserved. All rigid body motion in Euclidean space can be expressed in the composition of translation and rotation. Hereafter we deal only with three dimensional Euclidean space.

Consider a point $\mathbf{p}$ in 3D space whose coordinates are (p_x, p_y, p_z). A homogeneous vector representation[28] is given by

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Rotation is represented by a 3 x 3 matrix, part of special orthogonal group which have a determinant +1 [13]. If a rigid body is rotated by an angle θ on

each axis, rotation matrix for each case is given by

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Where $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$, represent rotation matrix on x, y, z axis respectively.

Rotation and translation together are represented as transformation matrix ($\mathbf{T}$), which is a 4 x 4 matrix of the form

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_3^T & 1 \end{bmatrix}$$

where $\mathbf{R}$ is a 3 x 3 Rotation matrix and $\mathbf{t}$ is 3 x 1 translation vector.

Inverse of transformation matrix is give by

$$\mathbf{T^{-1}} = \begin{bmatrix} \mathbf{R^T} & -\mathbf{R}^T\mathbf{t} \\ 0_3^T & 1 \end{bmatrix}$$

If a point $\mathbf{q}$ is a result of applying a rotation ($\mathbf{R}$) and translation ($\mathbf{t}$) on a point $\mathbf{p}$, it can obtained by

$$\mathbf{q} = \begin{bmatrix} \mathbf{R} & t \\ 0_3^T & 1 \end{bmatrix} \mathbf{p}$$

Point $\mathbf{p}$ can be transformed back given a rotation ($\mathbf{R}$) and translation ($\mathbf{t}$) to $\mathbf{q}$

$$\mathbf{p} = \begin{bmatrix} \mathbf{R^T} & -\mathbf{R}^T t \\ 0_3^T & 1 \end{bmatrix} \mathbf{q}$$

Consider this example where we register a data set $\boldsymbol{P} = \{p_1 ... p_{N_P}\}$ from a scanner on to a model $\boldsymbol{X} = \{x_1 ... x_{N_x}\}$. We have to find a transformation ($\mathbf{T}$) which minimizes the error between every transformed point in $\boldsymbol{X}' = \mathbf{T}\boldsymbol{P}$ to that in $\boldsymbol{X}$, which can be expressed as

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|x_i - \mathbf{T}p_i)\|^2 \tag{3.1}$$

This problem is referred to as Procrustes problem and explored widely in statistics.

Algorithm 3.1 Iterative Closest Point

Input: $\boldsymbol{X}$, $\boldsymbol{P}$, $\mathbf{T}_{\mathbf{initial}}$
Output: $\mathbf{T}$

```
1: new_error = MAX_VAL
2: T = T_initial
3: repeat
4:    error = new_error
5:    P' = transform(T, P)
6:    closest_points = return_closest_points(P', X)
7:    (T, new_error) = estimate_alignment(P, closest_points)
8: until (new_error − error) < threshold
```

3.2 The Iterative Closest Point Algorithm

ICP is used to align the 3D scans of an object with models. ICP primarily tries to minimize the difference between two given surfaces in an iterative way. We can summarize the classic ICP [6] in Algorithm 3.1.

ICP starts with assuming the closest points between the two 3D point sets as corresponding points and and that every point has a corresponding match. In the algorithm 3.1, *estimate_alignment()* estimates transformation between the two sets by least-square fitting. The data points are rotated with the new estimate and iterated over till a required error metric is reached. A good initial alignment estimate is a necessary condition for ICP to converge to a global minima efficiently. ICP is not robost againt outlier in the data set. A robot with a 3D scanner (for e.g. Kinect sensor) which is capable of producing dense 3D scans at a high rate, significant overlap between successive scans is possible in which case ICP is useful. For dealing with partially overlapping scenes with such 3D scanners, a simple distance threshold can be used to discard point correspondences. In general, ICP is very useful for a local registration problem, wherein we align successive scans.

If we can establish a precise point correspondences between few points in $\boldsymbol{P}$ to those in $\boldsymbol{X}$, we can find a transformation using a closed form solution and eliminate the iteration.

3.3 Closed Form Solutions

ICP requires a good estimate of an initial alignment, without which it may get trapped in local minima. Getting a good estimate is not trivial in many real world cases. In addition, a possibility of ICP not converging also exists. On the other hand, closed form solutions do not suffer from these, offer efficiency and robustness. Establishing point correspondences between two point sets is a primary problem. Keyframes from PTAM present a possible way of estab-

lishing 2D correspondences in the images which can inturn be translated to correspondences in point clouds, however, this is expensive in terms of computation and does not scale well. Placing markers (for e.g. ARToolKitPlus) to establish point correspondences is a viable option, keeping computation constraints in mind. Given a set of point correspondences, it is possible to determine the relationship between the two point sets, a problem referred to as *absolute orientation* [21] using a closed form solution.

A survey by [21] lists some of the major closed form solutions for and up to three dimensions which are based on:

- Singular Value Decomposition (SVD) [3]
- Unit Quaternions (UQ) [29]
- Orthonormal Matrices (OM) [30]
- Dual Quaternions (DQ) [76]

Authors present a through analysis of the algorithms and a comparison based on accuracy, robustness and stability. They concluded pointing that for all real world applications, these solutions offer the same level of stability and accuracy, with execution time being the only distinct factor among them. Given that LAPACK[47][38] is already used in the PTAM environment, obtaining SVD is simple. This made SVD based closed form solution an attractive approach.

The following steps shows how to obtain transformation using SVD algorithm ($\boldsymbol{P}$ to $\boldsymbol{Q}$) of two sets of 3D point correspondences $\boldsymbol{P}$ and $\boldsymbol{Q}$ of size N (here,$\boldsymbol{P}$, $\boldsymbol{Q}$ are 3 x N colum matrices).

1. Obtain centroids of the two sets. $p = \text{centroid}(\boldsymbol{P})$, $q = \text{centroid}(\boldsymbol{Q})$
2. Center the two point sets at the origin. $p_i \stackrel{\triangle}{=} \boldsymbol{P} - p$, $q_i \stackrel{\triangle}{=} \boldsymbol{Q} - q$
3. Obtain a 3x3 matrix H, $H \stackrel{\triangle}{=} \sum_{i=1}^{N} p_i {q_i}^t$
4. Compute SVD of H, $(U, V) = \text{SVD}(H)$
5. $X = VU^t$
6. if $determinant(X) = +1$, $\mathbf{R} = X$, $\mathbf{t} = q - \mathbf{R}p$
7. if $determinant(X)$ = -1, X is not pure rotation but also includes a reflection. In this case multiply third colum of matrix U by -1 and then compute X'. $\mathbf{R} = X'$, $\mathbf{t} = q - \mathbf{R}p$

3.4 Discussion

ICP is useful for local registration problems wherein we align successive scans or in situation where a good initial estimate is available. In literature, most ICP variants are tested with 3D data sets generated from laser scanner which are dense and very precise, however, this is not the case with PTAM where the 3D data is sparse and noisy. In addition, when robots are deployed from different regions and start mapping the environment, it may not be possible to obtain an initial alignment for the maps generated.

For establishing correspondence using keyframes, the views of the keyframes must first match. If the number of agents involved in mapping is large, finding a matching view among all the agents and then establishing correspondences by exhaustive search is a computationally expensive operation.

Aligning sparse point clouds which have very few point correspondences among them can be impossible without user intervention, unless those point correspondences are known. With the knowledge of point correspondences, transformation can be obtained using a closed form solution like the SVD based solution.

ARToolKitPlus marker system provides 4096 unique marker IDs and detecting the marker in keyframes is computationally inexpensive. By placing these markers in the environment to be mapped, precise point correspondences can be established between the two point clouds generated by robots using PTAM and can be aligned using a closed form solution.

Chapter 4

System Architecture and Implementation

With a visual SLAM like PTAM a quadrotor can navigate through a cluttered indoor environment and also create a sparse 3D world model of the environment. By deploying a swarm of quadrotors, we can map a large environment in parallel, to obtain greater details of the environment. The previous chapter introduces a possible way of aligning sparse world models generated from multiple quadrotors. With the knowledge of alignment, quadrotors can share their world models and localize each other on them. This chapter first presents a high level description of possible system architectures for mapping using multiple quadrotors and is followed by an implementation of a scheme.

4.1 System Architecture

In order to create a 3D map of an environment cooperatively, it is important to first create consistent local maps on each of the agents. PTAM can create a 3D map of feature points (FAST corners[59]) on each of the agents in the environment. These local maps can be aligned into a global map using the technique described in chapter *Aligning Point Clouds*, either centrally on a ground station or on agents themselves. An outcome of aligning these point clouds is that we can localize these agents on a global map.

The system consists of agents (quadrotors), which have a camera, additional sensors, communication capability and onboard computational resource. If agents have enough computational resource, they can have PTAM and ARToolKitPlus marker detection (ARTP) running onboard. The system includes a high-level controller (HLC) which is responsible for the motion coordination

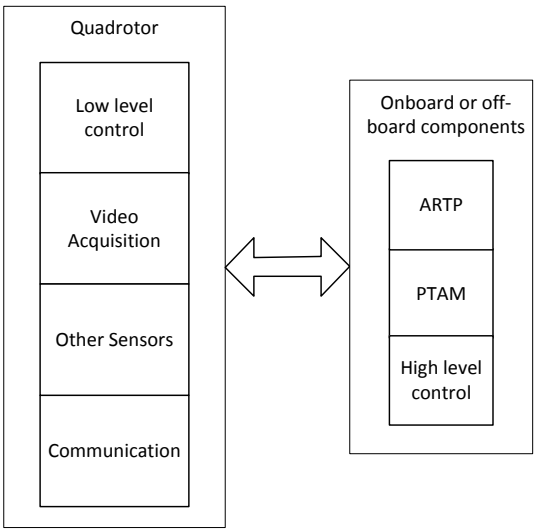


Figure 4.1: This figure shows the functional blocks of an agent

of the agents. An agent in the system can be decomposed into functional blocks as shown in Figure 4.1.

Based on these functional blocks the following configurations of the system are possible:

1. **Centralized coordination with PTAM off-board**: In this case, the agents mainly function as video source and PTAM is run off-board on a computationally powerful ground station. This architecture requires continuous high communication bandwidth since all agents are required to transmit video to ground station. Only enough computation resource is packed onboard the quadrotor to stabilize and control it. The marker detection (ARToolKitPlus) is also run on the ground station in tandem with PTAM to establish point correspondences and align the 3D maps. Agents can be easily coordinated from the ground station once these 3D maps are aligned. This architecture is suitable when number of agents in the system is low (two to three).

   In Figure 4.2, each of the quadrotors transmit their video stream over a wireless link to a ground station. On the ground station, multiple PTAM processes are instantiated and each handles a video stream to generate a 3D world model. The PTAM processes broadcast the point correspondences to each other when markers are detected. When all the markers in the environment are found one of the maps generated is choosen as a global map and all mappoints from other agents are incorporated into it. Here the agents are coordinated centrally by the

high-level controller running on the ground station.

2. **Centralized coordination with PTAM onboard**: The agents in this architecture have enough computational resource to run PTAM and ARToolKitPlus marker detector onboard. They only transmit 3D points from the map they build either as a steady stream or in bursts to the ground station. This requires only a fraction of bandwidth when compared to the previous case. When a marker is detected, point correspondences are established and transmitted to the ground station. Maps from all the agents are collected and aligned in the ground station. Coordination still takes place centrally on the ground station.

   Figure 4.3 shows PTAM algorithm running onboard the platform to generate 3D world model. The mappoints from world models on agents are transmitted to the ground station where one of the maps generated is used as a global map and mappoints from other agents are then incorporated into it when the transformations are computed. Since marker detection takes place on the quadrotors, alignment of maps on the ground station is suspended till markers are located by agents. On receiving point correspondences from all the agents the ground station computes the transformations of the maps from individual agents to the global map.

3. **Decentralized coordination with PTAM onboard**: It is also possible to eliminate the central component completely by having distributed algorithm for coordination running on these agents. Agents only need to broadcast their 3D point cloud and point correspondences established from the markers. This also requires minimum communication among the agents, which can be done in bursts, but needs to be coordinated. Each agent can incorporate the map of another simply by computing the transformation using the point correspondences and transform the point cloud to its reference frame. This architecture is highly scalable.

   Figure 4.4 shows the schematic representation of the decentralized coordination architecture. In Figure 4.4 each agent has enough computational power to run PTAM and create a world model. Agents can be deployed from different regions. Agents who come in communication range and have established a transformation can exchange world models and can cooperate with each other. Due to the distributed nature of coordination, agents can form groups based on either pre-planned directives or ad hoc when they come in communication range and cooperate to explore the environment.

The choice of architecture depends on the following factors:

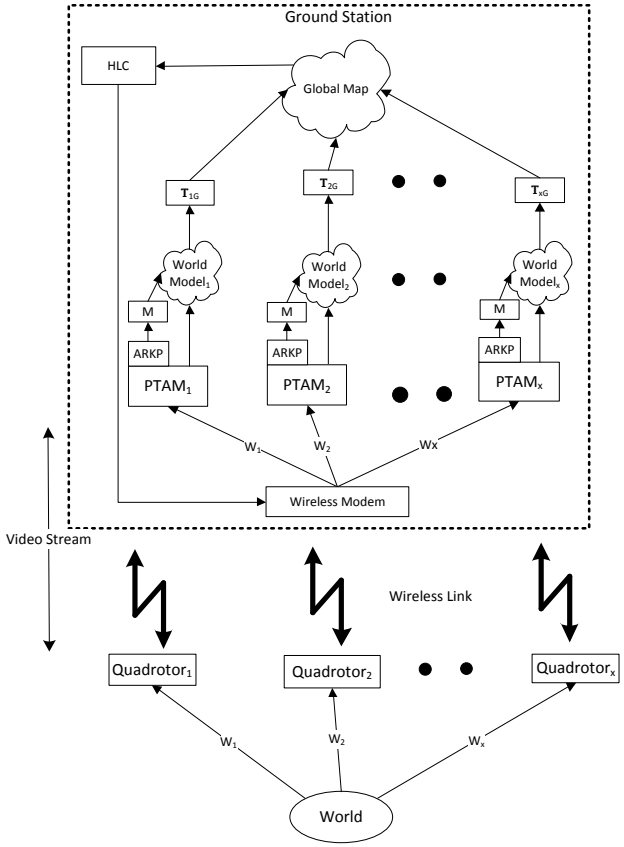


Figure 4.2: Schematic representation of centralized coordination with PTAM off-board

Communication Requirements

For a video stream which has a pixel resolution of 640x480 at 30 Frames Per Second (FPS) transmitting grayscale (8 bits per pixel) images without any compression, would require a bandwidth of 73.728 MB/s. A compression standard like JPEG can achieve an average compression of 5:1 [22] on a grayscale image (at the threshold of visible loss). With this compression, bandwidth requirement for the video stream can be brought down to 15-20 MB/s. A 3D point in the map is represented by a 3-tuple of floats (32 bits per float). A map with 147 frames would contain around 4000 mappoints which translates to 48 KB of data.

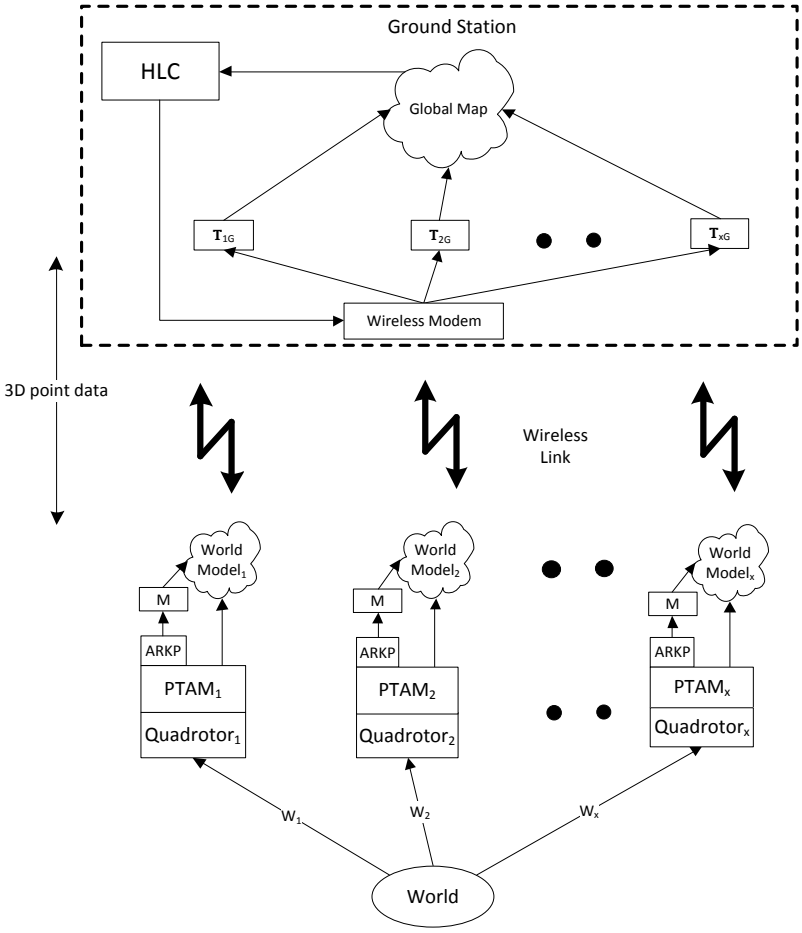


Figure 4.3: Schematic representation of centralized coordination with PTAM onboard

Computational Aspects

On a PC with Intel Core 2 Duo 2.66 GHz processor running Linux [33] with a video source of resolution 640×480, on an average tracking thread of PTAM takes 20ms for a frame (about 50 FPS). The computational effort in the tracking thread is proportional to the number of the features being tracked. By increasing the threshold in the FAST corner detection and the Shi-Tomasi score, we can control the number of features being tracked. The required Shi-Tomasi score and threshold of FAST corner detected for a given number of mappoints has to be computed empherically by extensive experimentation.

Increasing the pixel resolution of the video stream does not necessarily improve robustness of tracking. Increased pixel resolution certainly requires more

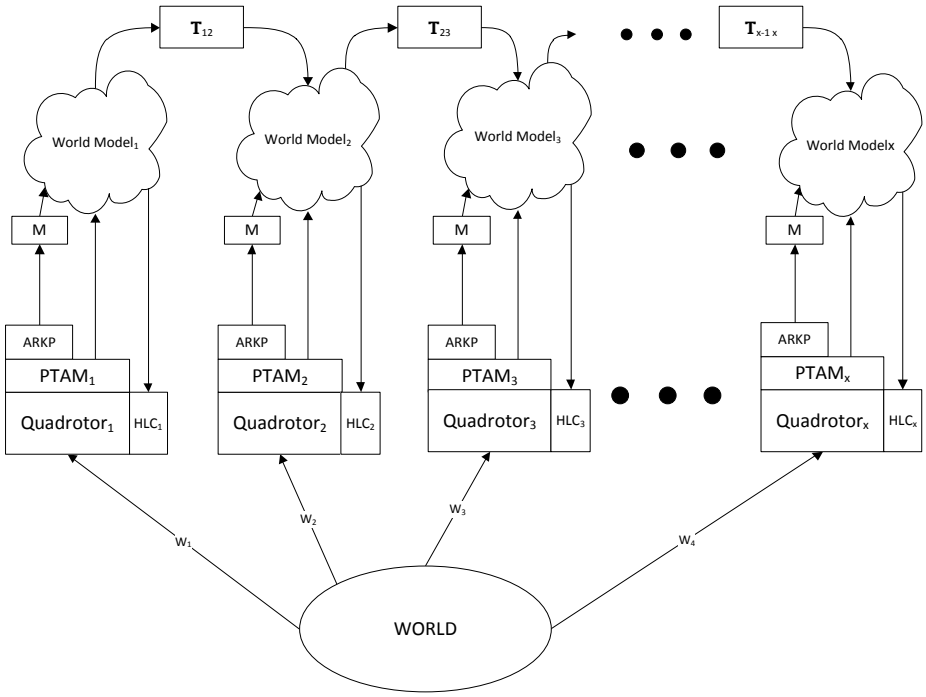


Figure 4.4: Schematic representation of decentralized coordination with PTAM onboard

effort in corner detection and requires more memory. Sub-sampling the video stream pixel resolution to 640x480 pixels and using this in PTAM can be an option. Enough features can be extracted with this pixel resolution for robust tracking. Another influence on computational requirement is the number of image pyramidal levels of a frame used by the tracking and mapping threads, by reducing the number of levels, we can bring down the computational requirement.

Mapping has two expensive operations, Local and global bundle adjustments. For a map with 100 to 149 keyframes, Local bundle adjustment takes 440ms and Global bundle adjustment takes 6.9s, while keyframe insertion takes about 40ms. ARToolKitPlus tracker requires 0.43ms on an average to detect a single marker. Global bundle adjustment operation's worst case computation complexity is $O(N^3)$, where N is number of keyframes in the map. Local bundle adjustment computational complexity also scales with map size, but at $O(NM)$, where M is the number of mappoints in the map. Figure 4.5 summarizes the communication versus onboard computational requirement for the three architectures.

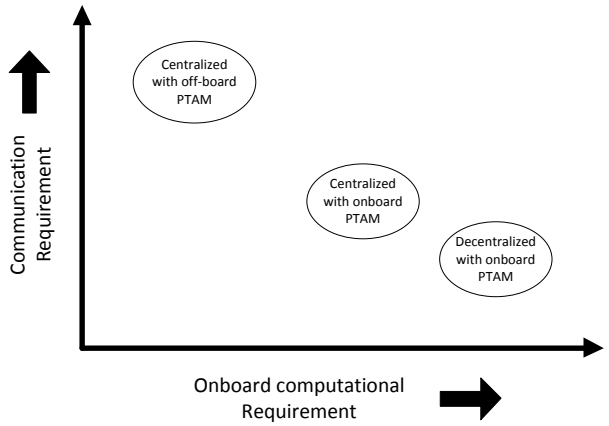


Figure 4.5: Communication vs Computation requirements for different configurations

Payload capability and onboard power of the platform

Payload capability of the platform is crucial as it influences other factors in the architecture. If the platform is capable of handling additional payload, only then we can add hardware onboard to increase available computing power. The platform should also be able to support the additional weight of the batteries required to power the additional hardware. The table 4.1 summarizes the comparison of the architectures.

4.2 Implementation

The centralized coordination with minimum onboard computational resource scheme has been implemented since the embedded computer onboard ARDrone is not powerful enough to run PTAM. In addition, the ARDrone has very limited payload capacity and onboard power.

The Figure 4.6 shows the setup used for experimentation. The ground station used is a dual core processor (1.86 GHz) computer which runs PTAM algorithm and the custom-made ARDrone application on a Linux operating system. The ARDrone application receives the video stream and sends attitude commands to the quadrotor. In addition, the ARDrone application can also log user commands and received sensor data into text files. The received frames from drone can also be saved as JPEG files.

Parallel Tracking and Multiple Mapping (PTAMM) [11] is used which is based on PTAM, but with additional features such as saving and loading maps and, it can also initialize multiple maps. Frames received from drone

Architecture	Centralized with off-board PTAM	Centralized with onboard PTAM	Decentralized coordination with onboard PTAM
Communication bandwidth	+++	++	+
Computing power onboard	+	+++	+++
Payload capability and onboard power	+	+++	+++
Scalability	-	++	+++

Table 4.1: A comparison of architectures

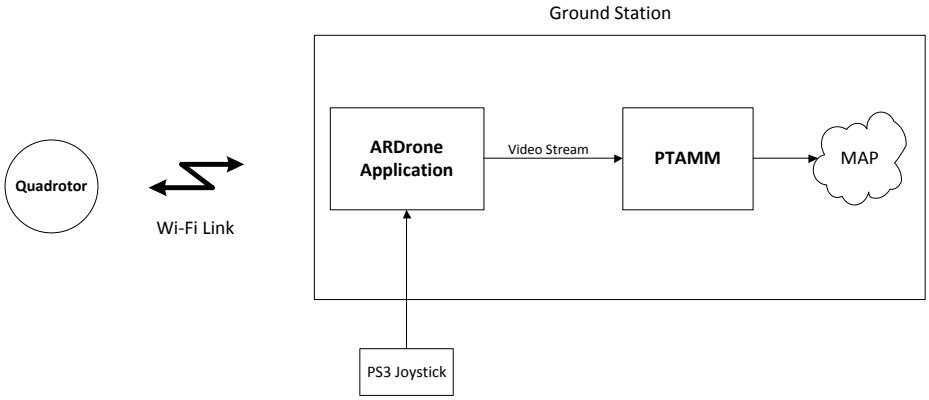


Figure 4.6: A schematic representation of the setup used for experimentation

are passed over to PTAMM by the ARDrone application via a shared memory which is synchronized using *named pipes*. From these frames the PTAMM creates the 3D world model. Results presented for aligning of world models from the agents were done offline as a proof of concept using *python scripts* to extract point correspondences from the saved maps and a matlab function which implements SVD solution to determine the transformation.

Modifications to PTAM algorithm

The video source used by Klein et al. differs from what is available to us, hence, modifications were necessary. The original algorithm required 640x480 pixels video source which allowed the algorithm to use four image pyramidal levels, the video stream from the quadrotor has a resolution of 320x240 pixels. The last pyramidal level would only contain 40x30 pixels, which will not have many features compared to other levels and mainly becomes a computational overhead. Hence, the number of image pyramid levels were reduced to three. Scaling up the image in pixel resolution will introduce artefacts which will have adverse effects on the map.

The ARDrone application, after being connected to the quadrotor over Wi-Fi starts receiving the video stream. A wrapper for PTAMM has been written to receive video frames from the ARDrone application. ARDrone quadrotor uses CMOS sensor with "rolling shutter" for image acquisition which introduces skewing, smearing, wobble effects and partial exposure. Rolling shutter is a method of image acquisition in which frames are recorded not as a snapshot of a single point in time, but rather by scanning across the frame either vertically or horizontally. On the contrary, global shutter captures the frame at a single instant of time.

It was noted that with this video source, number of features being tracked were far less than otherwise required, due to partial exposure effects (under fluorescent light). Hundreds of features are necessary for the tracking to be robust. The threshold for FAST corner detector at *Level* 0 was reduced to increase the number of features being tracked by the algorithm. In the original algorithm, a keyframe was added to the map only after twenty frames where the frame rate was 30Hz. The video stream from the quadrotor has a low frame rate (a little less than 15Hz), hence, this count is reduced to ten frames.

Establishing point correspondences between two maps

At least four markers should be placed in the environment to guarantee an initial alignment. All quadrotors should be able to locate these markers placed in the environment. BCH coded markers from ARToolKitPlus tracking library (4096 ids are available) can be used. First step in order to establish correspondences (called *anchor* points) between any *two maps* generated by PTAM is to establish correspondences between the marker and a map on an agent. A few strategies are discussed below.

PTAM uses FAST corners as interest points in an image. A possible strategy for establishing correspondence between two maps is to use vertices of the markers as anchor points. PTAM algorithm in the current implementation has been modified to detect features only in three image pyramidal levels where *Level* 0 is full resolution (320x240) of the frame. Figure 4.7 shows a keyframe of a map in which marker was detected and the red circles correspond

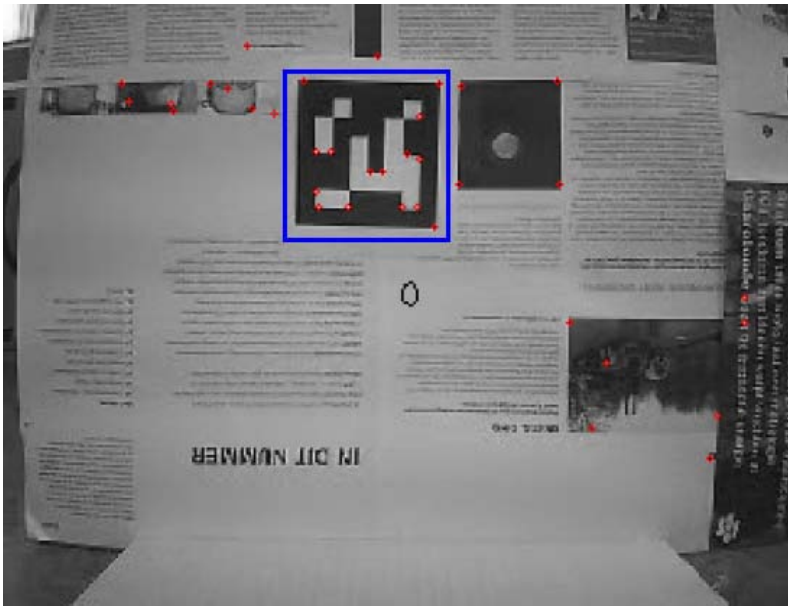


Figure 4.7: This figure shows a keyframe in which a marker is detected. The red circles represent the mapped features in this keyframe at *level* 0

to *level* 0 features which have been added to the map from this frame. The vertices of the markers are ideal candidates for FAST corner detector [59], due to the high contrast between black codeframe and the white paper. These corners also get a high Shi-Tomasi score [68]. If we eliminate the effects caused by rolling shutter, these corners are guaranteed to be mapped by PTAM when the marker is observed in the keyframe.

Figure 4.8 shows the relationship among the camera, maker and image used in ARToolKitPlus tracker library. ARToolKitPlus tracker library uses OpenGL coordinate system (right-handed system). Consider facing a computer screen, the top-left is the origin and top- left to right is positive X, top-left to bottom-left is positive Y and positive Z is away from the observer. Marker and camera use this coordinate system. For an image, same applies without the third dimension. PTAM uses the same image coordinate system, hence, results of marker detection from ARToolKitPlus tracking library can be used in PTAM. It is also assumed that quadrotor body frame coincides with that of camera frame (or the relationship is known).

When a frame is passed over to the mapping thread as a keyframe from the tracking thread, it can be checked for markers. In our context, ARToolKitPlus tracker is only used for marker detection and identification. ARToolKitPlus tracker internally stores vertices, edges, center of a detected marker (in image coordinates) in a structure named *ARMarkerInfo*. Figure 4.8 shows the vertices p_i in the image frame of the marker. This corresponds to vertex num-

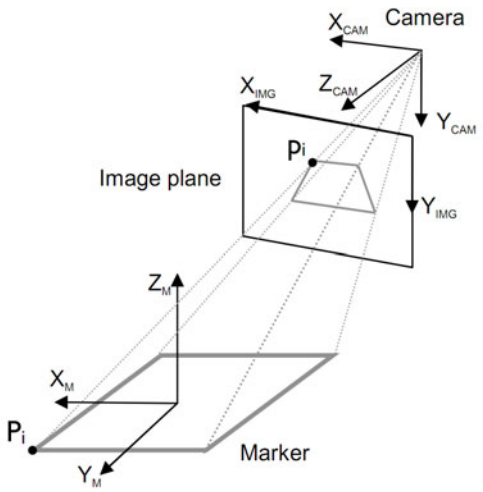


Figure 4.8: This figure shows the relationship among camera, marker and image coordinate systems [60]

ber 0 in the structure *ARMarkerInfo*. This structure is protected; it must be made public to access this information (this has to be done before compiling the ARToolKitPlus Library). This information is distortion corrected, which is essential for accurate pose estimation. Since we do not use the pose information, number of undistortion iteration is set to zero in the camera configuration file. The information in *ARMarkerInfo* can be used to locate and uniquely identify vertices of marker in the keyframe.

PTAM uses *KeyFrame* object to hold all the information about a keyframe required for both tracking and mapping thread. This information includes the image coordinates of all the mapped points in the keyframe. If a marker is detected in a keyframe, closest mappoints to vertices of the marker can be looked up with a tolerance (in pixel radius) from the mapped points and can be used as anchor points. This technique is used to obtain correspondences between marker and map. If two vertices are known, it is possible to estimate the position of the other vertices of the marker in the map. Function 4.1 summarizes the procedure. Function 4.1 is called whenever a new keyframe is inserted into the map. The keyframe is checked for marker, if the marker is present and confidence factor is more than 75%, we continue. Location of marker vertices in image coordinates are obtained from *ARMarkerInfo*. Function *find_closest_points()* parse through the keyframe data to obtain closest mappoints with image coordinates of the marker vertices. If all markers are found, the corresponding 3D coordinates of the mappoints are broadcasted to other PTAM processes.

PTAMM allows the user to save the state of the map to disk. This is used for aligning the maps offline. PTAMM saves a snapshot of keyframe information

function 4.1 Marker to map correspondence

```
1: function marker_to_map_corr(keyframe)
2: begin
3: (marker_id,confidence_factor)=ARToolKitPlus_detect_marker(keyframe)
4: if (confidence_factor> 0.75) then
5:    marker_vertices=ARToolKitPlus_MarkerInfo()
6:    anchor_points[marker_id][4]=find_closest_points(keyframe,
      marker_vertices)
7:    if (found_all_markers()& ∼broadcasted) then
8:       broadcast(anchor_points)
9:    end if
10: end if
11: end
```

in a XML file and keyframes as PNG files. A script was implemented to parse the XML and keyframe images to obtain the anchor points using function 4.1, but writes into a file instead of broadcasting. The XML file also holds the 3D point cloud data, which was exported as a CSV file and aligned in matlab.

Another way is to collect all the features detected inside marker's frame from all the keyframes in which a marker is detected. A normalized map (in image coordinates) of these features can be made and broadcasted. An agent 'A' can pick a set of features from the feature map of agent 'B' as anchor points for transformation from B to A.

Map maker in PTAM only adds most salient features in a keyframe to the map, which are first triangulated with an existing keyframe. It is possible to force the map maker to insert center of the marker (*ARMarkerInfo* holds this information) as a mappoint after triangulation. The 3D coordinates which correspond to center of markers can then be broadcasted as anchor points. This way we do not rely on map maker to pick the marker vertices.

Aligning sparse world maps

To obtain transformation between two 3D data sets using SVD, at least four corresponding points (anchor points) are needed. These anchor points are broadcasted when an agent locates markers and establish correspondences with their maps.

When an agent 'A' (PTAM process on the ground station) has at least four anchor points common with another agent 'B', agent 'A' can compute the transformation from B to A using the SVD algorithm from the previous chapter. Function 4.2 is called when an agent receives anchor points from another agent. Suppose that agent 'x' receives anchor points from an agent i, agent 'x' first updates its internal list of anchor points of other agents and then attempts to compute the transformation. If the agent 'x' has not found

function 4.2 Compute transform on agent 'x'

```
1: function compute_transformation(T_list, i, anchor_points)
2: begin
3: update_anchor_points(anchor_points_list, anchor_points)
4: if found_all_markers() then
5:    compute T_list[i] using SVD based algorithm
6: end if
7: return T_list
8: end
```

all markers, it attempts to compute the transformation after all markers are located. $\mathbf{T_{list}}$ is a datastructure which holds all the transformations from other agents to agent 'x'. When a map is selected as a global map, agents add their mappoints to the global map after selecting the transformation form $\mathbf{T_{list}}$.

Computational cost involved for obtaining transformation is least when compared to ICP and cost is nearly constant for a given number of point sets. If the anchor points are too noisy, an accurate transformation cannot be obtained, in which case RANSAC-like technique can be used to mitigate the effects of outliers [3].

Communication costs involved depend on the number of anchor points, each anchor point is 3-tuple of float requiring 12 Bytes and at least four anchor points are necessary. With the introduction of marker, an additional benefit is that we can also recover the scale of the map using the vertices of the marker, given that the distance between them is already known.

Chapter 5

Experiments and Results

In this chapter we evaluate performance of the ARToolKitPlus detection library and PTAM algorithm with the video stream from the onboard camera for observations from a single quadrotor. Two sets of point clouds that originate from different agents are aligned using markers to make a complete 3D world representation.

5.1 ARToolKitPlus marker identification range

Detection and identification of the marker is a crucial aspect of the system. We look at some of the factors which influence detection and identification of a marker by ARToolKitPlus library.

The marker

A marker consists of a frame enclosing a binary coded area. The binary data in the code region is encoded as black and white squares. During decoding of BCH code on the marker, the algorithm reduces the confidence factor (which starts with 100%) by 25% for every error bit that is detected and corrected.

Identification

To identify the marker correctly, the marker in the image must contain enough pixels which represent the coded area of the marker. Theoretically 6x6 pixels are sufficient, where each pixel represents a bit from the 36 bit binary code. The marker detection range can be increased by simply using a larger size marker and also by increasing the resolution of the camera.

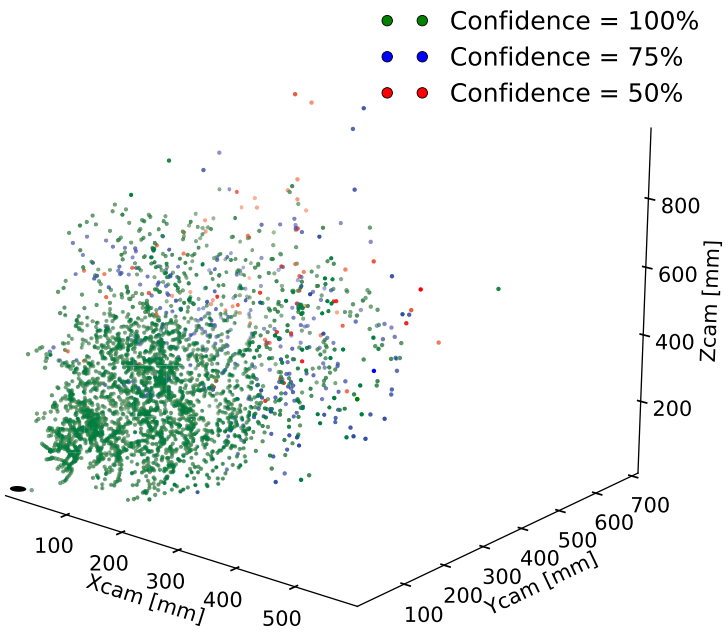


Figure 5.1: This figure shows confidence factor spread in first octant volume of camera coordinate system. The black circle at the origin represents the camera.

An empirical method is employed to determine the detection range of the marker with the onboard camera. Figure 5.2 shows a BCH marker of size 8x8 cm placed on a turntable, with Z_M of the marker coinciding with the rotation axis of the turntable. The camera (quadrotor) is moved up and down along the Z axis in the inertial frame and turned around X_{CAM} while the marker turns around Z_M such that the marker is kept inside the camera field of view (FoV). Refer to Figure 4.8 for clarity.

Results

Figure 5.1 shows the results in the camera coordinate system. The camera was calibrated to estimate intrinsic and extrinsic parameters using a Matlab Camera Calibration Toolbox [74]. The origin of the graph represents the center of the camera. Each dot represents the position where the marker was detected and the color representing the confidence factor. Since we take the internal working of ARToolKitPlus for granted, the focus is entirely on the confidence factor it provides. Only observations which have a confidence factor of at least 50% have been selected. This evaluation does not consider the effects of motion blur.

The results in Figure 5.3 provide a baseline for a 8x8 cm marker detection

Figure 5.2: This picture shows the test setup used in experiment 5.1. The marker is rotated at low speed while the quadrotor is moved.

in the video stream from the quadrotor which has a resolution of 320x240 pixels. The circular arc (about 350mm in radius) in Figure 5.3a shows the boundary up to which we observe maximum number of marker detections with a confidence factor of 100%.

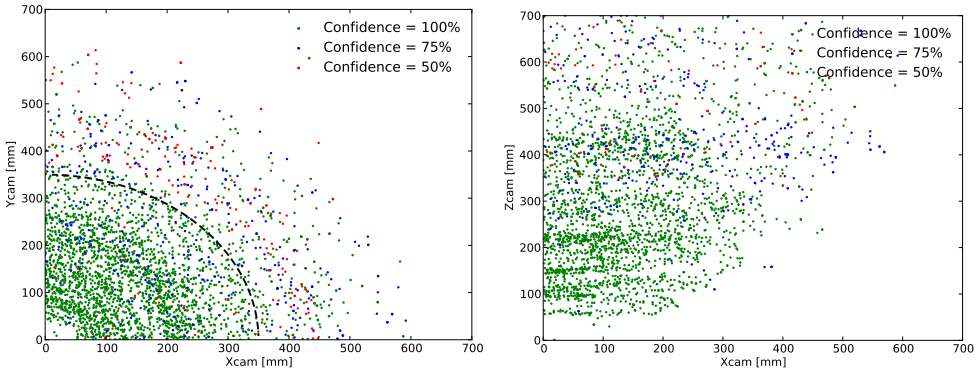
The identification of marker within this range is also influenced by the inclination of the marker plane and the camera image plane. As the angle of inclination between the two increases, the code area becomes too small to be sampled correctly and marker cannot be identified. Figure 5.4 shows the results of marker detection as angle of inclination is increased in steps of 10 degrees. The marker is placed at a distance of 35cm from the camera center.

Apart form the above, the rolling shutter of the CMOS sensor introduces skewing and wobble effects during motion which also adversely affect the marker detection. This can be eliminated by using a camera with global shutter [31].

5.2 Accuracy of mappoints

The rolling shutter introduces a bias in every step in the PTAM SLAM algorithm [34] (PTAM assumes a global shutter). This has a direct effect on the accuracy of the map.

The 2D image measurements of a feature taken on top of the frame tend to differ from the measurements of the same feature at the bottom of another frame due to the rolling shutter (when quadrotor is translating). Bundle adjustment refines the map by minimizing the re-projection errors of feature



(a) (b)

Figure 5.3: (a)XY view of Figure 5.1. The circular arc (about 350mm in radius) shows the boundary up to which we observe maximum number of marker detections with a confidence factor of 100%. (b) XZ view of Figure 5.1. The red, blue and green dots represent markers detected with confidence factors of 50%, 75% and 100% respectively.

observations. If the 2D image measurements are not corrected for rolling shutter effect, an accurate map cannot be created. It is not trivial to apply a correction since the speed of rolling shutter is unknown. We can minimize the effects of this by moving the quadrotor slowly or eliminate them by using a camera with a global shutter.

To estimate the accuracy of mappoints created with the setup, map of a flat surface having an area of 1.3 m^2 was created with $Z = 0$ plane of the map coinciding with the surface. Figure 5.5a shows a few point features of a mapped surface. The mean difference in the measured Z of the surface by PTAM in the complete map is 2.71 cm with a standard deviation of 3.51 cm. For mapping this scene, the camera was repeatedly moved around to that ensure all features are observed from multiple views. Most salient features were picked from the keyframes and actual positions of respective features were measured on the surface by hand. Figure 5.5 b, 5.5c and 5.5d show that PTAM map has a reasonable accuracy in $X-$ and $Y-$axes. We can see from Figure 5.5c and 5.5d that depth estimation of PTAM is noisy. The map starts to drift beyond $X_{map} = 30$cm in $Z-$direction, hence, deviations of upto 8cm can be seen among the points which are measured.

Another factor which affects map accuracy is global bundle adjustment not converging. As the quadrotor explores the environment, the map size grows. When the number of keyframes goes beyond a threshold, global bundle adjustment cannot keep up with exploration and is often aborted when a new keyframe is inserted. It would only converge when the quadrotor remains

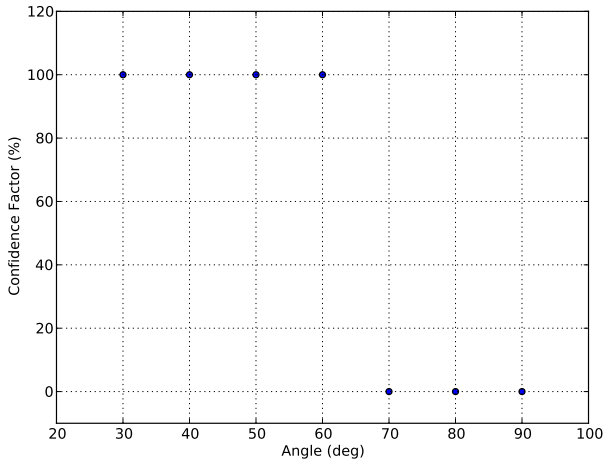


Figure 5.4: This figure shows confidence factor over angle of inclination. $X-$axis represents the angle of inclination between marker plane and camera image plane. The marker is placed at a distance of 35cm from the center of camera. We can see that beyond 60°, marker is unidentifiable.

stationary or is in an explored region for some time. The threshold depends on number of mappoints in the map and available computational power. The autonomous controller must consider these facts and must suspend exploration till the global bundle adjustment converges. An agent should not share its mappoints unless the map has converged. If computational limit has been reached i.e. the map is too large to converge the agent must return back to the base.

5.3 Factors affecting tracking in PTAM setup

PTAM can be used as a visual localization system onboard the quadrotor for autonomous flight which was demostrated by [8] with the camera facing downwards. For autonomous navigation of the quadrotor, position estimates from PTAM plays a curcial role. First the influence of the errors in the map on the position tracking accuracy is discussed and later some of the factors which affect tracking in PTAM are discussed.

Position tracking of the camera (quadrotor) in PTAM relies on the map built by the mapping thread. Errors in the map affect the position estimates of the camera in the inertial frame. A feature point in the environment can be more accurately triangulated by the mapping thread with multiple observations from different views.

In order to observe the tracking performance of PTAM, a flat surface was

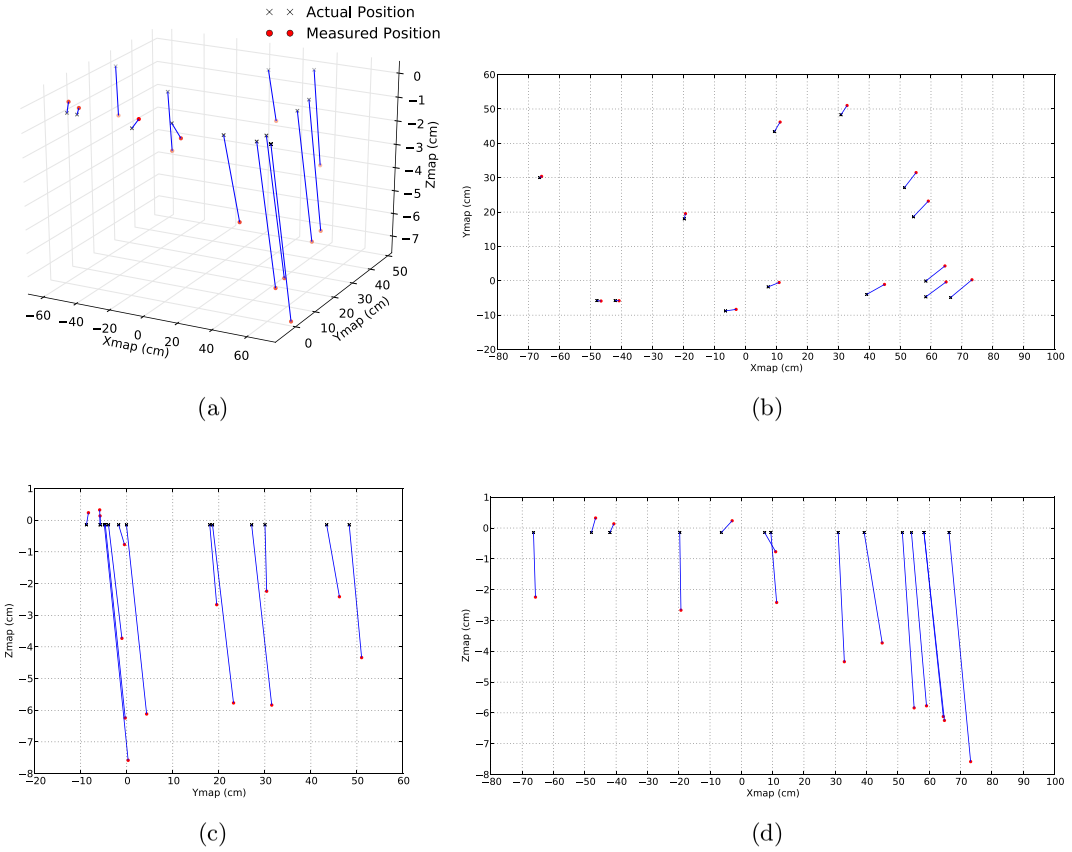


Figure 5.5: (a) 3D View of the mapped points. The lines indicate the error between measured and actual position. (b), (c) and (d) are XY, YZ and XZ views respectively. Notice that beyond $X_{map} = 30$cm the map starts drifting, hence, we observe large deviations.

mapped to create the map in Figure 5.6 by translating the quadrotor along the X_{map}. The $Z = 0$ plane of the map coincides with the surface. The quadrotor was placed on a cardboard box and translated to keep Y and Z constant. Since the feature points on the surface were observed only from one direction, in Figure 5.6 we can see a drift in the Z direction of the map. Since the map drifted in the Z direction we can also observe drift in Z position estimate in Figure 5.8c. It becomes necessary to observe an environment from multiple views in order to create a better map and suitable routines must be implemented on the quadrotor to loop through newly explored regions to minimize drift.

The tracking thread of PTAM indicates the quality (good, poor or lost) of tracking based on number of features that are being tracked in the current frame. Tracking and mapping go hand-in-hand, if tracking is lost, mapping is stopped. If the number of features being tracked falls below 50% (due to paratial occulsion of features/camera or change in lighting conditions), tracking is lost, in which case the quadrotor must maneuver back to a position where tracking is stable. If the quadrotor has drifted off too far from an explored region, tracking will be lost completely. The autonomous controller onboard the quadrotor should be designed in such a way so as to take inputs from the tracking thread and make sure quadrotor does not drift too far from the explored region. However, if the quadrotor has drifted off too far and cannot recover, the quadrotor can initialize a new map and start exploring the environment again.

For a setup requiring ground station, another issue which is likely to cause tracking failure is the Wi-Fi link. If a few frames are dropped because of CRC errors or the connection is lost for few frames, loss of tracking is inevitable. This is often the case with ARDrone quadrotor. The video stream freezes during take-off and during flight frames are often dropped due to CRC errors. If the quadrotor has drifted too far during this time, tracking will be lost. To mitigate loss of tracking due link failure, the onboard controller must hold the position of the quadrotor till the connection to ground station is re-established.

To make position estimates from the SLAM more precise, it is possible to augment visual estimates with estimates from an IMU using a filter[45]. It makes sense to take advantage of inertial sensing as they are already onboard the quadrotor for low level control.

5.4 Mappoint growth

The number of features being tracked directly reflects the computational effort in the tracking thread. By adjusting the threshold on the FAST corner detector, it is possible to control the number of features that are being tracked and added to the map.

Three scenes were mapped to observe the mappoint growth. Three maps

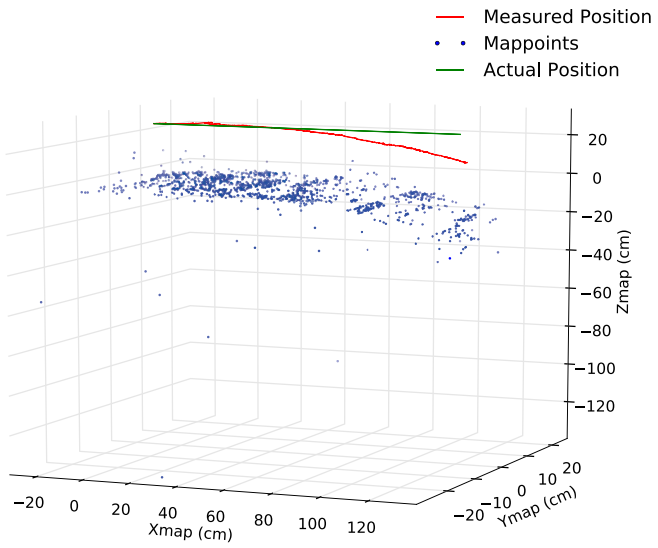


Figure 5.6: This figure shows a 3D view of the map. Notice that beyond $X_{map} = 60$cm the map starts to drift

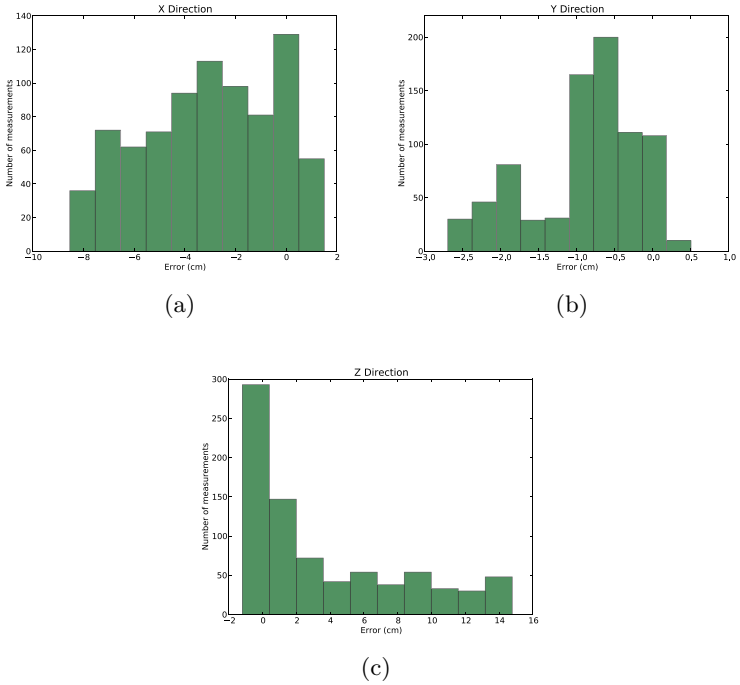
1, 2 and 3 represent areas of 8 m^2,1.3 m^2 and 1.9 m^2 respectively. Figure 5.9 shows the growth of mappoints which have been added to the map. Another map (Map 4) of the third scene was created, but with threshold [59] of FAST corners relaxed from 10 to 5 at level 0, hence Map 4 shows an accelerated growth of mappoints.

A computationally expensive operation in the mapping thread of the PTAM is the local and global bundle adjustment. By having stricter heuristics in keyframe insertion and by reducing the features being added to the map, we can essentially map a larger area before reaching the computational limit.

Using a wide angle camera can also help in reducing the number of keyframes required to describe a scene. In the implementation of [77], a downward looking camera with a 150 degree fish-eye lens is used and this setup can map an area of at least 50 m^2 in size.

5.5 Aligning Maps

Due to inherent shortcomings of the onboard camera, large surfaces could not be mapped. Two sets of small scenes were mapped and then aligned for this



(a) (b)

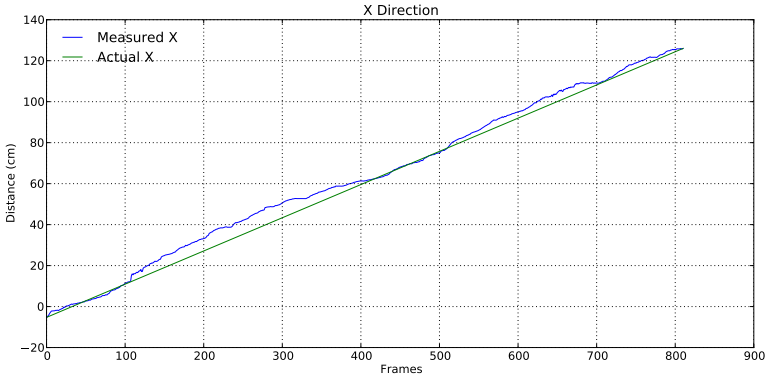
(c)

Figure 5.7: (a), (b) and (c) show distribution of errors in X , Y and Z direction respectively.

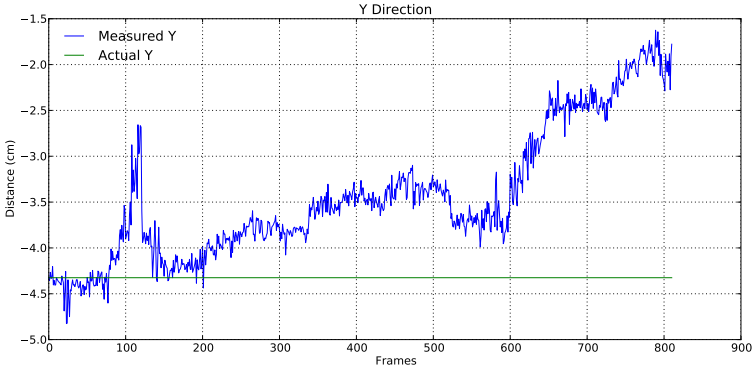
experiment. Note that newspapers are placed on the surfaces to give rich texture, the setup is unable to map otherwise. Markers of size 8x8cm are used for alignment of scenes. The quadrotor was moved roughly around 35 cm from the surface to enable marker detection.

A table of area 1.3 m^2 as seen in Figure 5.11a was mapped on quadrotors A and B. Four markers are placed on the table surface. Both maps are initialized such that $Z = 0$ plane of the maps coincide with table surface. Figure 5.12 shows the two maps in their reference frames. In order to create a more accurate map of feature points, the table was traversed slowly multiple times from different views. Out of the 90 keyframes in the map of quadrotor A, 15 keyframes had all markers with vertices. In map B, out of 106 keyframes, 17 keyframes had all markers with vertices. With anchor points from markers, the map of quadrotor B is aligned with that of quadrotor A in Figure 5.13. Notice in Figure 5.14a map of quadrotor B has drifted in Z direction.

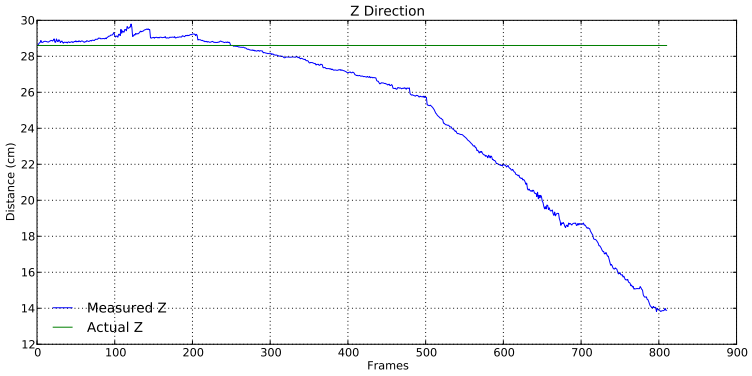
In a different setup, two scenes of area 1.6 m^2 (white board) and 0.81 m^2 (desk) shown in Figure 5.15a and 5.15c were mapped on two agents A and B respectively. The purpose of this experiment was to show that alignment is possible with minimum overlap between the scenes. The desk sequence map



(a)



(b)



(c)

Figure 5.8: (a), (b) and (c) show errors in position estimates in $X-$, $Y-$ and $Z-$direction respectively. Notice the drift in the $Z-$direction this is due to accumulation of errors in depth perception.

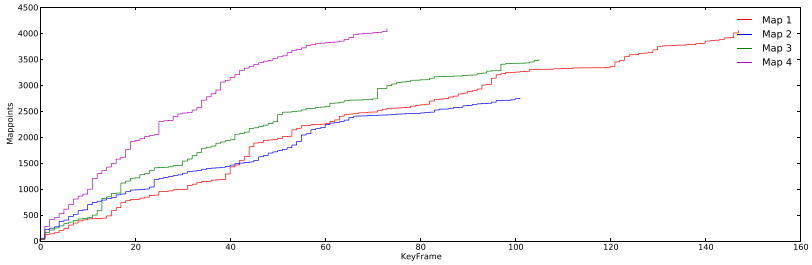
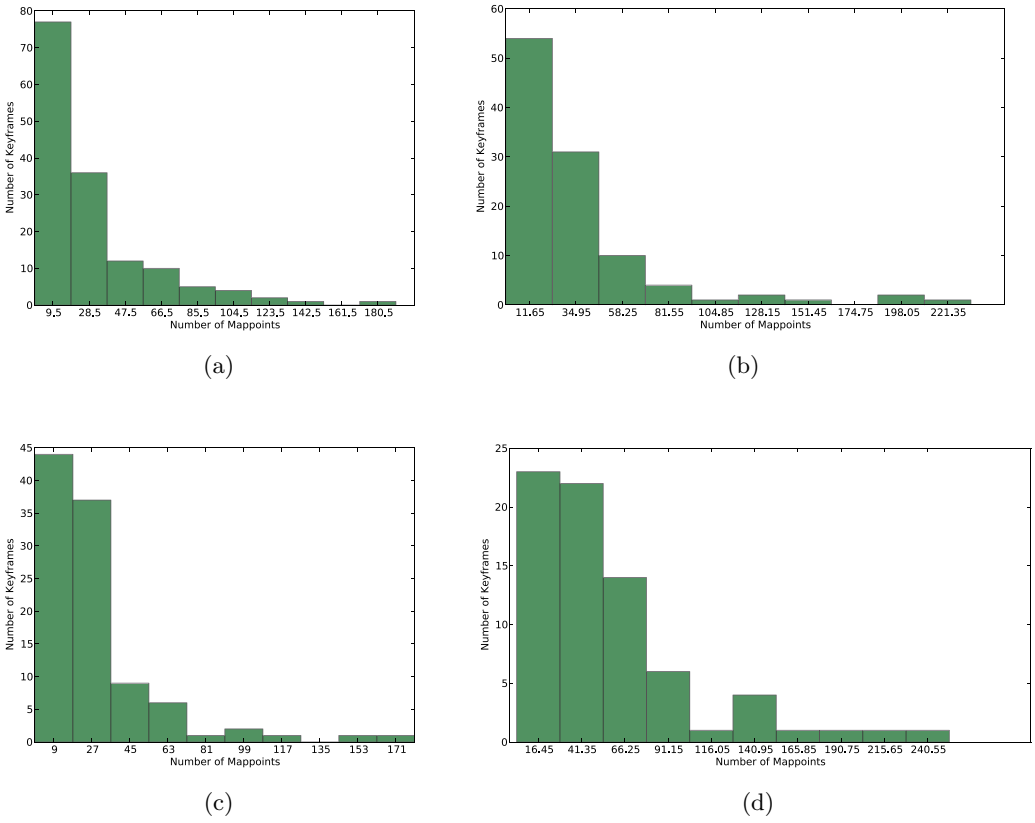


Figure 5.9: Mappoint Growth



(a) (b)

(c) (d)

Figure 5.10: (a), (b), (c) and (d) show the distribution of mappoints in the keyframes of the four maps respectively. Ideally most keyframes must provide an average of 30 mappoints. This will allow us to map larger area before reaching the computational limit. Due to shortcomings of the camera we see that most keyframes in the maps have only around 10 mappoints. Very few mappoints in the map can lead to unreliable tracking.

was initialized such that the desk surface coincides with $Z-$axis of the map. While the board sequence was initialized such that wall coincides with $Z-$axis of the map. These maps were aligned with two markers as seen in Figure 5.15e. The desk sequence contains 119 keyframes, in which 7 keyframes with marker vertices were identified. The white board sequence has 120 keyframes, in which 6 keyframes were identified with marker vertices. Transformation from map B to A was computed using anchor points from the markers and results of the alignment are presented in Figure 5.16. In Figure 5.15e shows the overlap between the scenes where the markers were placed.

5.6 Extension to 'N' maps

From Figure 5.11 and 5.16d we see that alignment is possible with the proposed approach. We were able to align sparse point clouds that overlap completely and partially. We first discuss the results of the two test cases and next we see how this technique can be applied in a cooperative scenario.

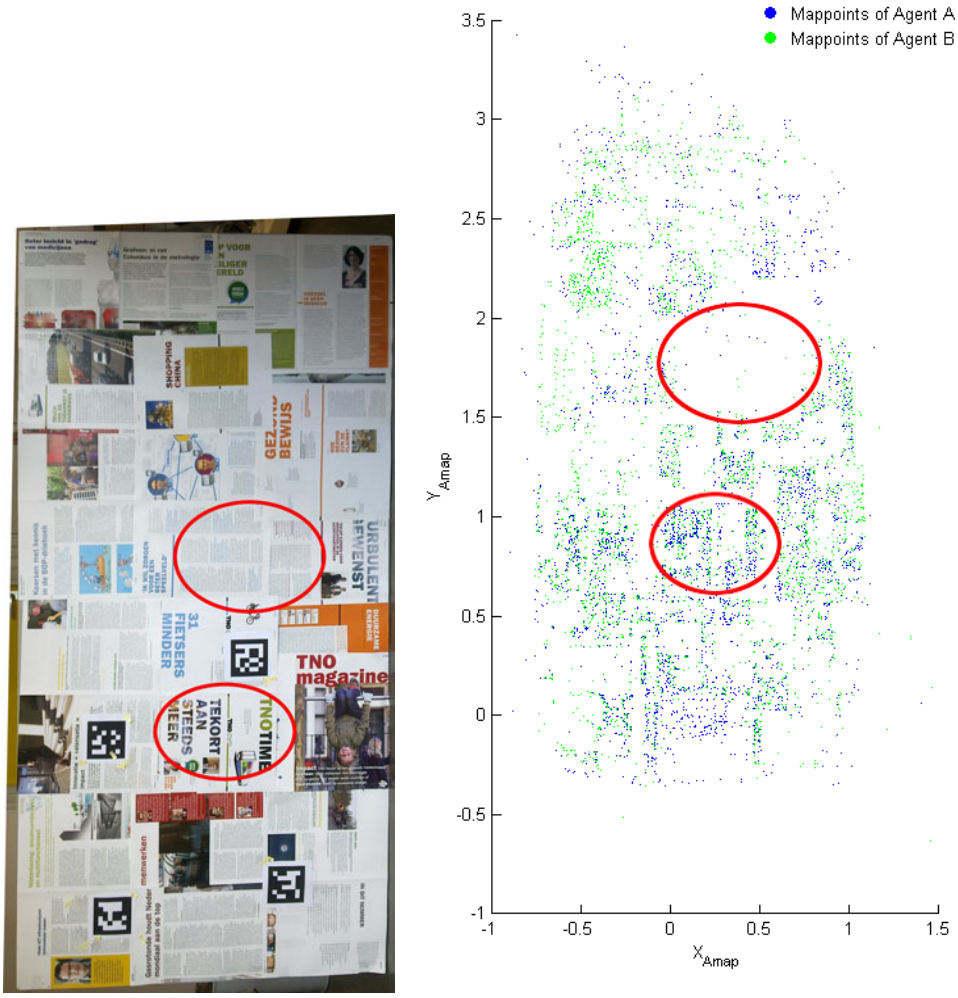
Markers of adequate size must be placed in the environment for successful detection. For a successful alignment, the marker must be detected, identified and vertices of the marker must be captured as mappoints. The table scene in Figure 5.11a was mapped several times and at least four marker vertices out of sixteen marker vertices (four markers) were detected 100% of the time. A primary reason for failure to detect marker vertices is due to the blurring of keyframes. This can be minimized by using a camera with a global shutter.

Notice in Figure 5.14 and 5.16d that the maps were noisy and alignment was still possible. Errors in the transformation can occur due to a noisy anchor point. Figure 5.18 shows alignment results when one of the four anchor points used for computing transformation was moved by 20% of its original position in $X-$, $Y-$ and $Z-$directions. If the estimated coordinates of anchor points are not accurate, obtaining a precise transformation between the maps may not be possible.

The 3D coordinates of the anchor point in a map can be erroneous due to errors in triangulation (due to noisy observations) of marker vertices by PTAM. The effects of outliers on transformation can be mitigated by using RANSAC technique with SVD based closed form solution. Erroneous anchor points of a single marker can be eliminated by checking if they lie on the same plane and form a square within a tolerance.

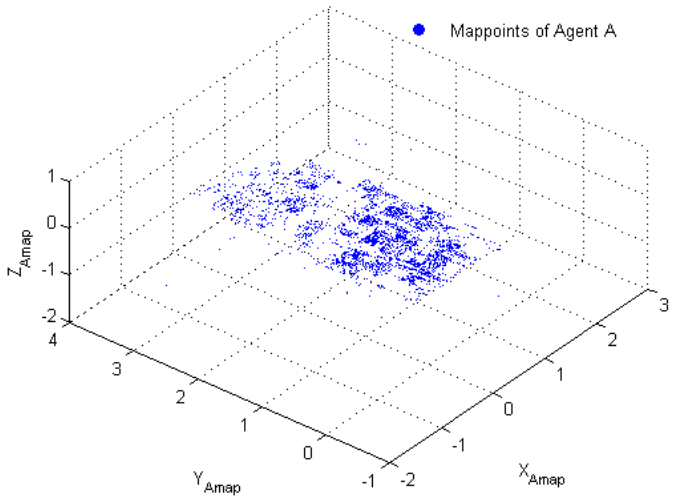
To compute transformation using SVD based solution takes 0.414 ms for 100 anchor points on the ground station. Computational cost for obtaining the transformation using SVD based solution is constant for a given set of anchor points and is not influenced by size of the maps.

Any number of maps can be aligned, given that we have enough number of anchor points common among them. Consider that we have 'N' agents in the system with a decentralized architecture. Now suppose that agents

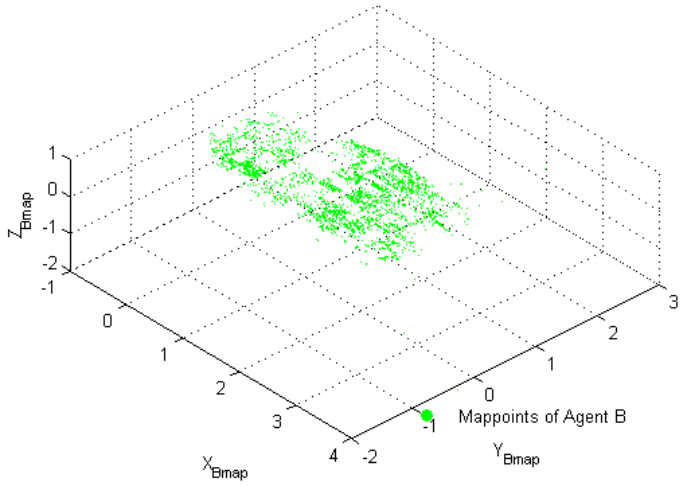


(a) (b)

Figure 5.11: (a) shows the scene which has been mapped (b) is the top view of the aligned maps (Not to scale). Notice the highlighted features. The empty spaces in the maps are due to the small text which is too fine for the setup to map, however, visibly larger text has been mapped.



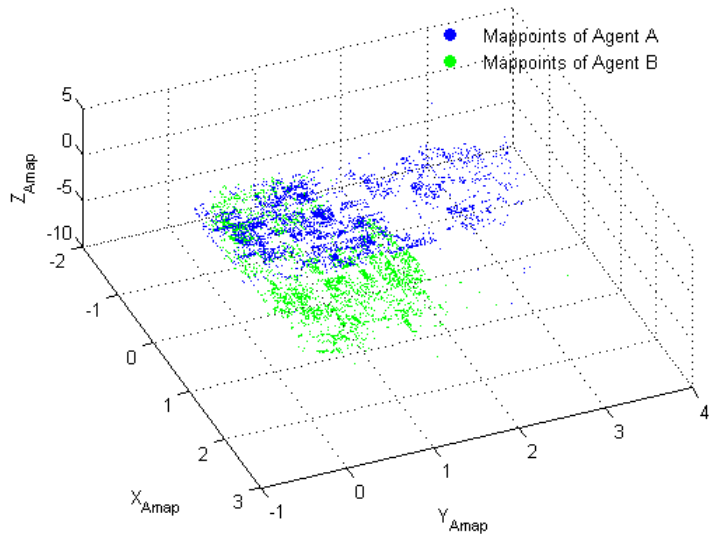
(a)



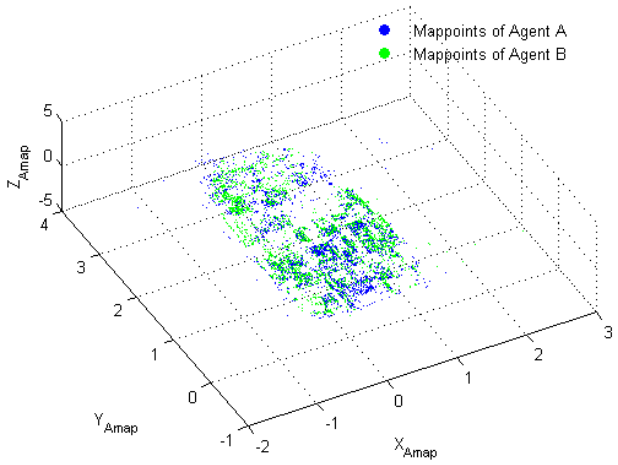
(b)

(c)

Figure 5.12: (a) and (b) view of mappoints in their reference frame (Not to scale). Notice that maps have been initialized at opposite ends of the table. (c) shows the setup used for mapping the table sequence. The quadrotor was moved roughly around 35cm above the table.

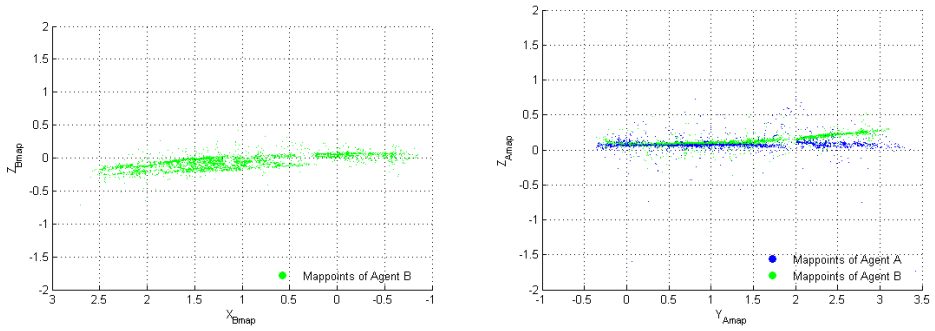


(a)



(b)

Figure 5.13: (a) To demonstrate the results when the mappoints of agent B are combined with that of agent A's without transformation. (b) Shows the view of map A after computing the transformation to align the reference frames. Map A is used as the base (global map) for combined point clouds. (Not to scale)



(a) (b)

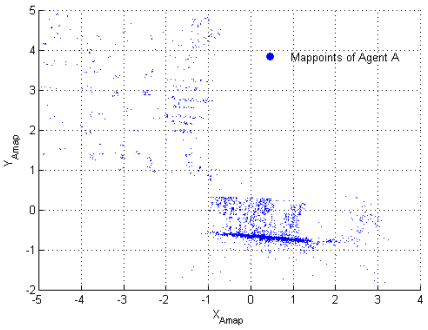
Figure 5.14: (a) shows XZ view of map B. Note the drift in map in $Z-$direction. (b) shows YZ view of the aligned maps of table sequence. The region on the left contains the markers, hence, both maps align well in that region. The map B may have drifted global, but it is still consistent locally where the markers are, therefore, alignment was still possible. (Not to scale)

have located the markers and are able to compute transformation to each other. When the transformation matrices between maps are known, agents can incorporate mappoints from others onto their map.

The camera (quadrotor) position is computed with respect to the reference frame of the agent by the tracking thread. When the maps are aligned, position estimates of an agent can be interpreted by other agents. Agents only need to broadcast their current position estimates from their tracking thread for the participating agents to localize each other and the high level controllers on the agents can use this information to improve the map further.

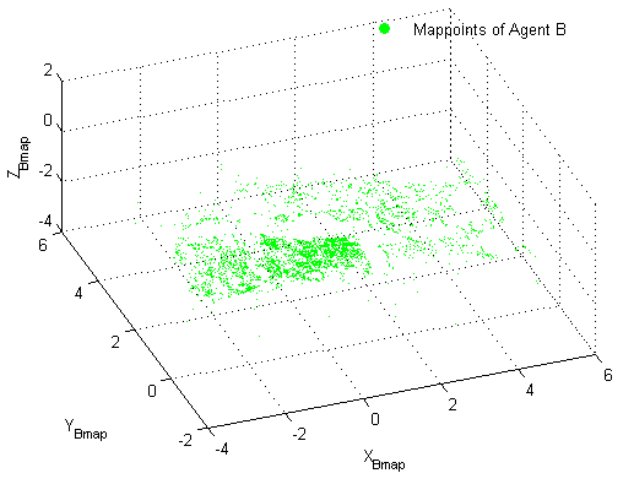
A key advantage of using markers to align in the case of decentralized architecture is that agents dont need to exchange complete map information to align or localize each other on their maps. The only disadvantage is that we have to spread markers in the environment. This can be accomplished by a leader quadrotor, who first flies into the environment dropping the markers.

(a)



(b)

(c)



(d)

(e)

Figure 5.15: (a) shows the board which has been mapped to produce (b). (c) shows the desk scene mapped to produce (d). (e) shows the region of overlap between the two scenes. (Not to scale)

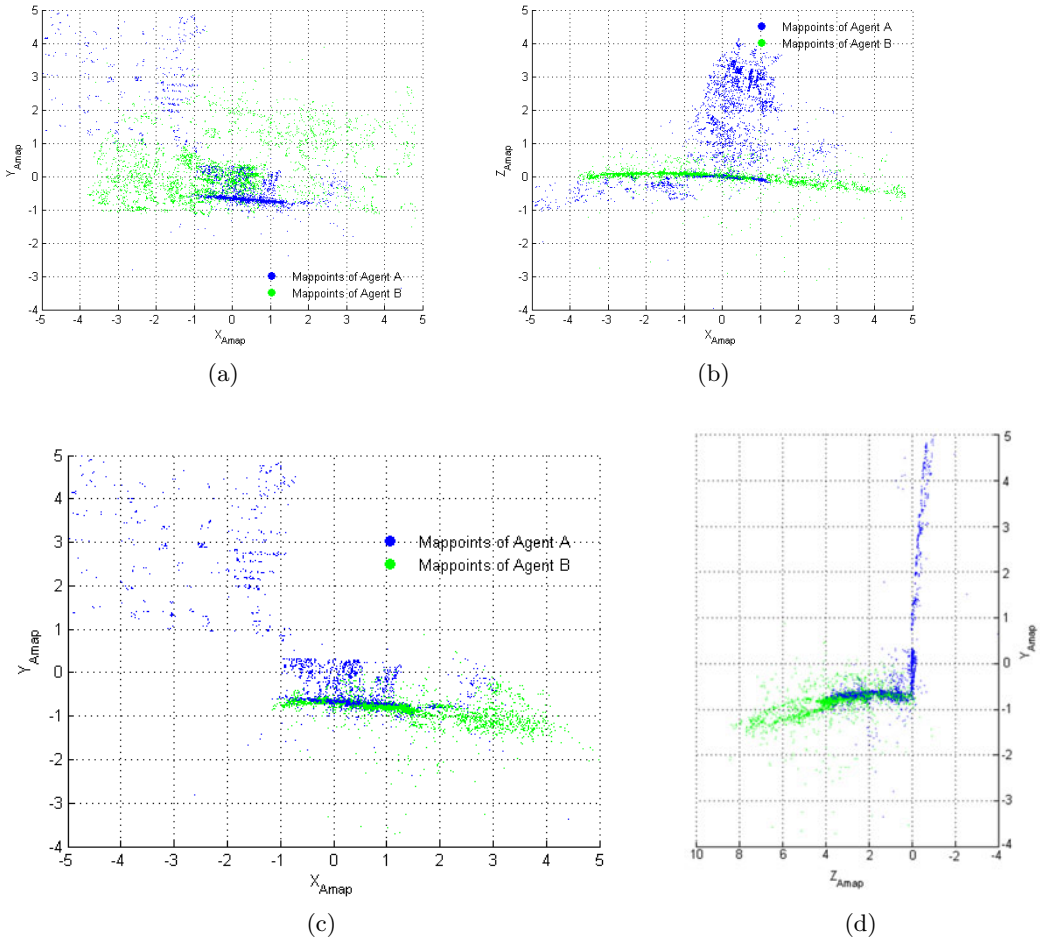
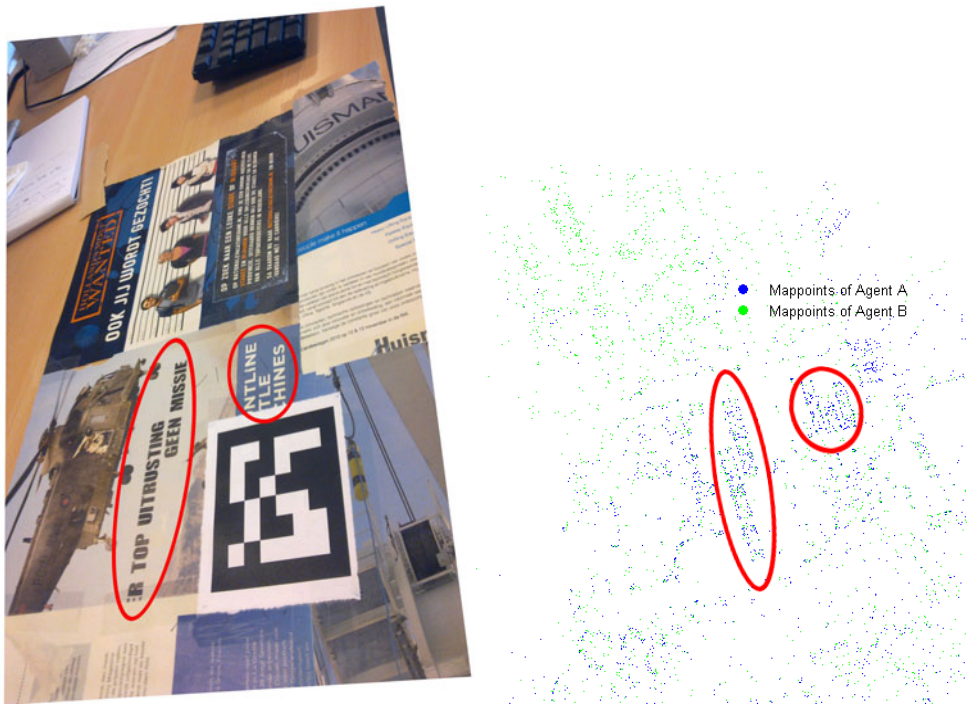


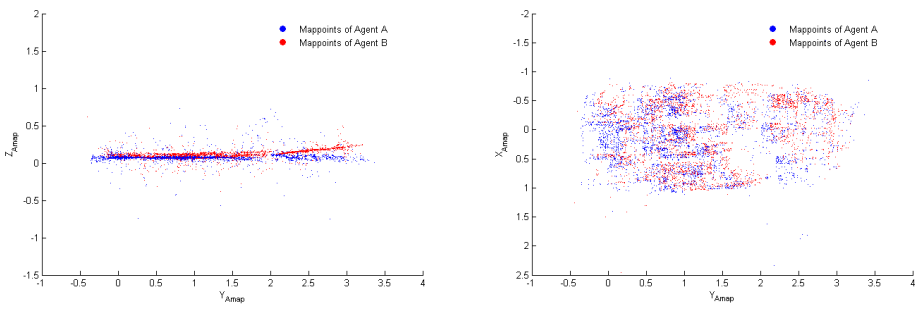
Figure 5.16: (a) and (b) show the map of agent A when mappoints of agent B are added without transformation. (c) and (d) shows the aligned maps, here the map of A is used as a base (global map) for combined point clouds. Notice how the desk lines up next to the board in (d), refer Figure 5.17c for the scene. (Not to scale)



(a) (b)

(c)

Figure 5.17: (a) view of the overlapping scene in desk/board sequence, highlighted with some features. (b) shows close up of the aligned map. Notice that highlighted features from both the maps align. (c) shows the desk and board from a different view.



(a) (b)

Figure 5.18: (a) XY and (b) XZ view of the mis-aligned maps of table sequence when one of the anchor point was moved by 20% in X ,Y and Z directions. (Not to scale)

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Cooperative indoor 3D mapping is still in its nascent stage. Previous work on cooperative indoor 3D mapping have focused on generating indoor 3D maps on land-based robots using occupancy grids. In literature much of the work on 3D mapping using a quadrotor has been done using expensive platforms and specialized sensors. Research on cooperative indoor 3D mapping using multiple quadrotors hasn't been explored yet.

Research in real-time monocular Simultaneous Localization and Mapping (SLAM) is gaining momentum in the computer vision community and is a new paradigm in 3D mapping using robots. This makes the camera probably the cheapest and compact 3D map building sensor available. However, more research is needed on the use of monocular SLAM for Unmanned Aerial Vehicle (UAV) autonomous navigation and mapping of an environment.

In this thesis we have explored the possibility of using a monocular SLAM Parallel Tracking and Mapping (PTAM) for cooperative 3D mapping of an environment with a fleet of quadrotors. We have demonstrated a proof of concept with a simple scenario wherein agents were able to combine each others' observations successfully in chapter 5.

A novel use of fiducial markers for alignment of sparse point clouds generated from a monocular SLAM is proposed in this thesis. This method requires minimum computations and from the results in chapter 5 we observe that it is successful. Alignment of point clouds is done analytically and does not require a more expensive numerical approach, as shown in chapter 3. Only by sharing the anchor points agents can establish the geometric relationship between the maps. This technique can be used to align sparse 3D point cloud from any

keyframe-based visual SLAM algorithm.

Under this framework three architectures are proposed for cooperative mapping using monocular SLAM PTAM. Among the proposed architectures, two are centralized and one is a decentralized architecture. The proposed architecture centralized coordination with PTAM off-board can cope with a platform which has minimum onboard computational power and payload capability, here the maps are built off-board and coordinated centrally from the ground station. In another centralized architecture, the maps are built on the agents but are coordinated centrally from the ground station. In the decentralized architecture, maps are built on the agents and are coordinated with a distributed algorithm.

We have evaluated the performance of the PTAM and ARToolKitPlus marker tracking system with the video stream from an inexpensive quadrotor with experiments in chapter 5. The results from chapter 5 show that the onboard camera is a severe handicap for the platform. In addition, the video stream is also found to freeze during take-off and frames were often dropped during flight. A recommendation for further research with this platform would be to use a camera with global shutter on a separate video stream link. Compressing images onboard the quadrotor and then transmitting can also ease the bandwidth requirement for better frame rate. However, compression and decompression will introduce delays which have to be accounted for.

With respect to the research questions, we can conclude that:

- *Can 3D mapping be achieved using a low cost quadrotor?*
  Yes, it is possible with a centralized architecture proposed in chapter 4. However, the above recommendations on camera must be considered.

- *How are sparse noisy 3D world models aligned?*
  Using fiducial markers sparse 3D worlds models generated by a visual SLAM can be aligned in computationally efficient manner as discussed in chapter 3.

- *What are the possible system architectures for cooperative mapping?*
  Three architectures which address aspects such as communication, (computational) power and payload capability of the platform are described in chapter 4.

A general concern in this system is the use of markers to establish mappoint correspondences. The role of the marker is that of “visual beacons”, which enables establishing point correspondences between two point clouds. The question would be how feasible is it to place markers in an environment to be mapped. Having a leader robot to deploy markers in places where agents enter the structure can be an option. In applications such as inspection, embedding markers into structures can also be an option.

6.2 Future Work

Cooperative robotics encompass many research areas and it is not possible to address all the issues within the time span of this master thesis. Some issues which are yet to be addressed are cooperative motion coordination and task planning (high-level controller) for further optimally mapping the environment after the initial alignment of the maps. Apart from this, the following can be considered as future work:

- A primary focus in thesis has been establishing a geometric relationship between sparse point clouds in a computationally efficient manner. This has been extended to a cooperative scenario where the agents can share their maps. Another possibility for cooperative mapping using a visual SLAM would be the use of *a priori* map. If we can guarantee relocalization, *a priori* map could be used by all agents and cooperate to improve it with the help of markers in the environment. Accurate and guaranteed relocalization of a quadrotor with a sparse distribution of keyframes is challenging and this can be an interesting research question.

- Our approch for aligning point clouds is very effective when there is no initial estimate available and ICP performs well for fine alignments when the point clouds are relatively aligned. A hybrid approch can be expected to give an excellent alignment. More experiments needs to be performed in this direction.

- The functional blocks in this thesis were glued together with pieces of code. A better approach can be to use a meta-operating system like Robot Operating System (ROS) [58]. However, PTAM is yet to be ported into ROS. Step one would be to port PTAM to ROS.

- In this work we have used a SLAM intended for single robot. A truly distributed visual SLAM algorithm is yet to be explored. The visual SLAM assumes the world to be unchanging and cannot deal with a dynamic world, more research is required for mapping a dynamic world.

Bibliography

[1] M Achtelik, M Achtelik, S Weiss, and R Siegwart. Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[2] Parrot ARDrone. `http://ardrone.parrot.com/parrot-ar-drone/uk/`.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. (5):698–700, 1987.

[4] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. Range - robust autonomous navigation in gps-denied environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1096 –1097, may 2010.

[5] A. Bachrach, S. Prentice, R. He, and N. Roy. Range - robust autonomous navigation in gps-denied environments. *Journal of Field Robotics*, 2011.

[6] P.J. Besl and H.D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239 –256, feb 1992.

[7] P. Biber, H. Andreasson, T. Duckett, and A. Schilling. 3d modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3430 – 3435 vol.4, sept.-2 oct. 2004.

[8] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision based mav navigation in unknown and unstructured environments. In *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, pages 21–28, 2010.

[9] R. C. Bose and D. K. Ray Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, Mar. 1960.

[10] T. Brotherton, R. Luppold, P. Padykula, and R. Wade. Generic integrated phm/controller system. In *Aerospace Conference, 2005 IEEE*, pages 3427 –3437, march 2005.

[11] R. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proc. 12th IEEE Int. Symp. Wearable Computers ISWC 2008*, pages 15–22, 2008.

[12] Huahua Chen and Zezhong Xu. 3d map building based on stereo vision. In *Networking, Sensing and Control, 2006. ICNSC '06. Proceedings of the 2006 IEEE International Conference on*, pages 969 –973, 0-0 2006.

[13] John H. Conway. *Atlas of Finite Groups: Maximal Subgroups and Ordinary Characters for Simple Groups*. Oxford University Press, USA, Jan. 1986.

[14] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052 –1067, june 2007.

[15] G.C.H.E. de Croon, K.M.E. de Clercq, R. Ruijsink, Remes B., and C. de Wagter. Design, aerodynamics, and vision-based control of the delfly. *International Journal of Micro Air Vehicles*, 1:71–97, June 2009.

[16] Gksel Dedeoglu and Gaurav Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *DISTRIBUTED AUTONOMOUS ROBOTICS SYSTEMS*, pages 251–260. Springer-Verlag, 2000.

[17] The ARDrone SDK developers guide. `https://projects.ardrone.org/embedded/ardrone-api/index.html`.

[18] M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17:229–241, 2001.

[19] I. Dryanovski, W. Morris, and Jizhong Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1553 –1559, oct. 2010.

[20] E. Eade and T. Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469 – 476, june 2006.

[21] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl.*, 9:272–290, March 1997.

[22] JPEG FAQ. `http://www.faqs.org/faqs/jpeg-faq/part1/`.

[23] M. Fiala. Artag, a fiducial marker system using digital techniques. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 590 – 596 vol. 2, june 2005.

[24] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.

[25] Dragon Flyer. `http://www.draganfly.com/our-customers/`.

[26] Ana Irene Ramrez Galarza and Jos Seade. *Introduction to Classical Geometries*. Birkhuser Basel, ISBN: 9783764375171, first edition, 2002.

[27] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. Towards a navigation system for autonomous indoor flying. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[28] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[29] Berthold K P Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, 1987.

[30] Berthold K. P. Horn, H.M. Hilden, and Shariar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOURNAL OF THE OPTICAL SOCIETY AMERICA*, 5(7):1127–1135, 1988.

[31] iPS uEye. `http://www.ips-vision.com/products_ids.php#1`.

[32] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 85–. IEEE Computer Society, 1999.

[33] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. 6th IEEE and ACM Int. Symp. Mixed and Augmented Reality ISMAR 2007*, pages 225–234, 2007.

[34] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando, October 2009.

[35] GRASP Labs. `https://www.grasp.upenn.edu/`.

[36] Hokuyo UTM-30LX Laser. `http://www.hokuyo-aut.jp`.

[37] Gim Hee Lee, Markus Achtelik, Friedrich Fraundorfer, Marc Pollefeys, and Roland Siegwart. A benchmarking tool for mav visual pose estimation. In *Proc. 11th Int Control Automation Robotics & Vision (ICARCV) Conf*, pages 1541–1546, 2010.

[38] libCVD computer vision library. `http://mi.eng.cam.ac.uk/~er258/cvd/`.

[39] ARToolKitPlus Pose Tracking Library. `http://studierstube.icg.tugraz.at/handheld_ar/artoolkitplus.php`.

[40] James J. Little, Cullen Jennings, and Don Murray. Vision-based mapping with cooperative robots. In *in Proceedings of SPIE - Sensor Fusion and Decentralized Control in Robotic Systems*, pages 2–12, 1998.

[41] MicroDrone. `http://www.microdrones.com/index-en.php`.

[42] Mikrokopter. `http://www.mikrokopter.de/ucwiki/en/FlightCtrlManual`.

[43] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[44] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505, 2010.

[45] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *J. Intell. Robotics Syst.*, 61:287–299, January 2011.

[46] ARToolKitPlus on Launchpad. `https://launchpad.net/artoolkitplus`.

[47] Linear Algebra PACKage. `http://www.netlib.org/lapack/`.

[48] L.E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220 –240, apr 1998.

[49] Lynne E. Parker. Current state of the art in distributed autonomous mobile robotics. In *Distributed Autonomous Robotic Systems*, pages 3–12. Springer, 2000.

[50] PixHawk. `http://pixhawk.ethz.ch/micro-air-vehicles/`.

[51] ARDrone API project. `https://projects.ardrone.org/`.

[52] Mark Pupilli and Andrew Calway. Real-time camera tracking using a particle filter. In *In Proc. British Machine Vision Conference*, pages 519–528, 2005.

[53] Nageswara S. V. Rao. Terrain model acquisition by mobile robot teams and nconnectivity. In *In Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000 - this volume*, 2000.

[54] J. Rekimoto. Matrix: a realtime object identification and registration method for augmented reality. In *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*, pages 63 –68, jul 1998.

[55] Jun Rekimoto and Yuji Ayatsuka. Cybercode: designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*, DARE '00, pages 1–10, New York, NY, USA, 2000. ACM.

[56] MIT Robust Robotics Group. `http://groups.csail.mit.edu/rrg/index.php?n=Main.VisualOdometryForGPS-DeniedFlight`.

[57] Rui Rocha, Jorge Dias, and Adriano Carvalho. Cooperative multi-robot systems:: A study of vision-based 3-d mapping using information theory. *Robotics and Autonomous Systems*, 53(3-4):282 – 311, 2005.

[58] Robot Operating System (ROS). `http://www.ros.org/wiki/`.

[59] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[60] P. Rudol, M. Wzorek, and P. Doherty. Vision-based pose estimation for autonomous indoor navigation of micro-scale unmanned aircraft systems. In *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, pages 1913–1920, 2010.

[61] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145 –152, 2001.

[62] J. Ryde. *Cooperative 3D mapping and Localisation of Multiple Mobile Robots.* PhD thesis, University of Essex, 2008.

[63] J. Ryde and H. Hu. Laser based simultaneous mutual localisation for multiple mobile robots. In *Proceedings of IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 404–409, Niagara Falls, Canada, July 2005.

[64] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2024 –2030, dec. 2006.

[65] Kinect Sensor. `http://www.xbox.com/en-GB/kinect`.

[66] sFly: Swarm of Micro Flying Robots. `http://www.sfly.org/doku.php/`.

[67] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[68] Jianbo Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593 –600, jun 1994.

[69] R. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty, 1986.

[70] J. Stuehmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition (Proc. DAGM)*, pages 11–20, Darmstadt, Germany, September 2010.

[71] Ascending Technologies. `http://www.asctec.de/home-en/`.

[72] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 321 –328 vol.1, 2000.

[73] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005.

[74] GML MatLab Camera Calibration Toolbox. `http://graphics.cs.msu.ru/en/science/research/calibration/matlab`.

[75] Daniel Wagner and Dieter Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Computer Vision Winter Workshop 2007*, 2007.

[76] Michael W. Walker, Lejun Shao, and Richard A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54(3):358 – 367, 1991.

[77] Stephan Weiss, Markus Achtelik, Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. Intuitive 3d maps for mav terrain exploration and obstacle avoidance. *J. Intell. Robotics Syst.*, 61:473–493, January 2011.

[78] R. Zask and M.N. Dailey. Rapid 3d visualization of indoor scenes using 3d occupancy grid isosurfaces. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 02, pages 672 –675, May 2009.