VISION BASED COLLABORATIVE LOCALIZATION AND PATH PLANNING FOR MICRO AERIAL VEHICLES

A Dissertation

by

SAI HEMACHANDRA VEMPRALA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Srikanth Saripalli
Committee Members,	Swaminathan Gopalswamy
	Sivakumar Rathinam
	Dylan Shell
Head of Department,	Andreas A. Polycarpou

May 2019

Major Subject: Mechanical Engineering

ABSTRACT

Autonomous micro aerial vehicles (MAV) have gained immense popularity in both the commercial and research worlds over the last few years. Due to their small size and agility, MAVs are considered to have great potential for civil and industrial tasks such as photography, search and rescue, exploration, inspection and surveillance.

Autonomy on MAVs usually involves solving the major problems of localization and path planning. While GPS is a popular choice for localization for many MAV platforms today, it suffers from issues such as inaccurate estimation around large structures, and complete unavailability in remote areas/indoor scenarios. From the alternative sensing mechanisms, cameras arise as an attractive choice to be an onboard sensor due to the richness of information captured, along with small size and inexpensiveness. Another consideration that comes into picture for micro aerial vehicles is the fact that these small platforms suffer from inability to fly for long amounts of time or carry heavy payload, scenarios that can be solved by allocating a group, or a swarm of MAVs to perform a task than just one. Collaboration between multiple vehicles allows for better accuracy of estimation, task distribution and mission efficiency.

Combining these rationales, this dissertation presents collaborative vision based localization and path planning frameworks. Although these were created as two separate steps, the ideal application would contain both of them as a loosely coupled localization and planning algorithm. A forward-facing monocular camera onboard each MAV is considered as the sole sensor for computing pose estimates. With this minimal setup, this dissertation first investigates methods to perform feature-based localization, with the possibility of fusing two types of localization data: one that is computed onboard each MAV, and the other that comes from relative measurements between the vehicles. Feature based methods were preferred over direct methods for vision because of the relative ease with which tangible data packets can be transferred between vehicles, and because feature data allows for minimal data

transfer compared to large images. Inspired by techniques from multiple view geometry and structure from motion, this localization algorithm presents a decentralized full 6-degree of freedom pose estimation method complete with a consistent fusion methodology to obtain robust estimates only at discrete instants, thus not requiring constant communication between vehicles. This method was validated on image data obtained from high fidelity simulations as well as real life MAV tests.

These vision based collaborative constraints were also applied to the problem of path planning with a focus on performing uncertainty-aware planning, where the algorithm is responsible for generating not only a valid, collision-free path, but also making sure that this path allows for successful localization throughout. As joint multi-robot planning can be a computationally intractable problem, planning was divided into two steps from a vision-aware perspective. As the first step for improving localization performance is having access to a better map of features, a next-best-multi-view algorithm was developed which can compute the best viewpoints for multiple vehicles that can improve an existing sparse reconstruction. This algorithm contains a cost function containing vision-based heuristics that determines the quality of expected images from any set of viewpoints; which is minimized through an efficient evolutionary strategy known as Covariance Matrix Adaption (CMA-ES) that can handle very high dimensional sample spaces. In the second step, a sampling based planner called Vision-Aware RRT* (VA-RRT*) was developed which includes similar vision heuristics in an information gain based framework in order to drive individual vehicles towards areas that can benefit feature tracking and thus localization. Both steps of the planning framework were tested and validated using results from simulation.

DEDICATION

To my parents

and my grandfather

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my advisor, Dr. Srikanth Saripalli. My grad school experience has been absolutely amazing and I thank Sri wholeheartedly not only for his tremendous research and academic support, but also for providing an incredible work environment, many exciting opportunities throughout the course of my PhD and for being a wonderful advisor in every way possible. I would also like to thank the rest of my PhD committee: Dr. Dylan Shell, Dr. Swaminathan Gopalswamy and Dr. Sivakumar Rathinam for the excellent discussions, their insightful comments and suggestions through the development of this work, and many crucial remarks that helped shape my work as well as this document.

Furthermore, I would like to especially show my gratitude to the late Dr. Alberto Behar. I have had the honor of working closely with Alberto on several incredible projects during my time at Arizona State University, and his broad and diverse knowledge, infectious enthusiasm and passion for scientific exploration influenced my mindset greatly. Working under his guidance on robots that explored the most remote parts of this planet helped nurture my interest in the field of autonomous robotics, which eventually brought me all the way to a PhD.

I consider myself to be lucky to have had the opportunity to be part of two schools during my PhD years. Many thanks go to the members at the School of Earth and Space Exploration at the Arizona State University: professors I have had the pleasure of interacting with, and other friends I made during the first few years of my PhD. All the members of ASTRIL, my former lab at ASU, deserve special thanks for being wonderful colleagues and close friends. After the transition to Texas A&M University, I have had the privilege of being one of the first few members of the Unmanned Systems Lab, and a chance to make new friends. Many thanks go to the members of the Unmanned Systems Lab as well as the CAST group at TAMU: for their help with my experiments, for the illuminating whiteboard

discussions and for being great friends and colleagues.

Last but not the least, an immense amount of gratitude goes to my parents for their belief in me throughout and for always encouraging me to follow my dreams. I will forever be indebted to them for their unbelievable amounts of support through many challenging times.

CONTRIBUTORS AND FUNDING SOURCES

This work was supported by a dissertation committee consisting of Dr, Srikanth Saripalli [advisor], Dr. Swaminathan Gopalswamy and Dr. Sivakumar Rathinam of the Department of Mechanical Engineering and Dr. Dylan Shell of the Department of Computer Science. Graduate study was supported by a research assistantship from Texas A&M University.

NOMENCLATURE

UAV	Unmanned Aerial Vehicle
MAV	Micro Aerial Vehicle
VCL	Vision based Collaborative Localization
VCP	Vision based Collaborative Path-planning
PNP	Perspective N Point
SFM	Structure From Motion
CMA-ES	Covariance Matrix Adaptation - Evolution Strategy
NBV	Next Best View
NBMV	Next Best Multi-View
RRT	Rapidly Exploring Random Tree
VA-RRT*	Vision Aware Rapidly-exploring Random Tree

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The field of aerial robotics encompasses the study of a variety of flying machines, with the main goal of operation without need for direct human intervention. These machines are usually known as Unmanned Aerial Vehicles (UAV). The American Institute of Aeronautics and Astronautics defines a UAV as follows:[1]

"an aircraft which is designed or modified to not carry a human pilot, and is operated through electronic input initiated by the flight controller or by an onboard autonomous flight management control system that does not require flight controller intervention."

In the current age, unmanned aerial vehicle technology is on the cutting edge of robotics research. Recently developed UAV platforms recently possess various perception and navigational capabilities combined with the level of decisional autonomy required to execute complex tasks. Naturally, UAVs present a unique advantage over ground robots because of their capability to exploit the third dimension, and the ability to move in six degrees of freedom which makes them capable of navigating cluttered and/or challenging environments. Within the classification of unmanned aerial vehicles, a certain class of vehicles known as 'rotary-wing' unmanned aerial vehicles (RW-UAV) deals with vehicles that are specifically tailored to achieve hovering ability. This enables the vehicles to examine environments without having to be in motion continuously, such as for instance, reaching a vantage point to image or map their surroundings. RW-UAV in turn have various classifications, ranging from helicopter type configurations that were examined initially, to the later-developed multi-rotor configurations. Multi-rotor configurations currently enjoy immense popularity both as research platforms as well as hobby flying vehicles usually due to their small size and maneuverability, along with a relative ease of prototyping.

Multirotor UAV currently enjoy widespread popularity in several application domains. The most well known ones are aerial photography, filmmaking, precision agriculture etc., with others being actively developed such as utilization within search and rescue

applications, surveying, reconnaissance and structural inspection.

1.1 Micro Aerial Vehicles

Micro aerial vehicles are a special class of UAV that have gained general attention as well as research focus in the past few years. Galiski and Zbikowski[2] define a micro aerial vehicle as a vehicle "small enough to be practical for a single-person transport and use." In a more colloquial sense, the term MAV is typically used for platforms that are under 1m of size in any dimension. While the phrase MAV can be used to refer to different classes of flying vehicles, throughout this dissertation, it is used exclusively to refer to multi-rotor configurations, which are more commonly referred to as quadrotors, hexrotors etc. depending on the number of actuators/rotors the vehicle is equipped with.

Multirotor micro aerial vehicles have a common physical configuration: where irrespective of number, all rotors are aligned in a plane, and all the rotor axes are perpendicular to that plane. A multirotor is also an inherently unstable configuration: while it affords higher agility, it creates numerous challenges in terms of control, perception and planning when it comes to implementing autonomous operations.

Due to the requirement of small size, MAVs are usually equipped with small and low-power sensors. The choice of sensor typically involves many considerations and for MAVs, monocular cameras have been found to be a good fit as they share many advantages with the vehicles themselves, such as inexpensiveness and small size. The fidelity of information is what determines the accuracy of localization and thereby other functions related to autonomy: and even in this regard, monocular vision sensors have great potential due to their richness of information assuming sufficient illumination. Other vision sensors that have the potential to produce richer information have certain limitations: stereo cameras, for instance, would have limitations on the distance between the two cameras which in turn limits their ability to perceive depth. For these reasons, monocular cameras are almost ubiquitous on both commercial and research MAV platforms today.

1.2 Challenges associated with MAV

The small size of MAVs coupled with the instability of the multirotor platform creates numerous challenges for autonomous operations. Some of the relevant ones are discussed below.

1.2.1 *Limited power and payload*

The constraint of small size on MAVs translates directly to a low payload carrying capacity. As MAVs have to continuously expend energy in order to stay airborne, the amount of vertical thrust spent in order to hover scales up with the weight of the equipment onboard. Naturally, this means MAVs are limited not in terms of sensors and computers, but also with regards to the size of batteries, which in turn reduces flight time. Hence, MAVs cannot usually make use of heavy, power-hungry sensors although they may provide much more accurate information for the purpose of localization. Lighter, low-power sensors are usually favored, albeit at the expense of low-frequency, noisier or less accurate data in general. The algorithms that form the autonomy pipeline for MAVs need to take this into account and need to possess an additional degree of robustness to cope with the possibly low quality data, while also ensuring low computational complexity. MAVs can only handle low power computational modules due to the same reasons as before, and this is another constraint that needs consideration when designing autonomy algorithms.

1.2.2 *Degrees of Freedom*

As discussed earlier, one of the inherent advantages of MAVs is that they can navigate in full six degrees of freedom (3D space). In contrast, ground vehicles are usually limited to two degrees of freedom in position and one as a yaw angle: which makes state estimation and control significantly more complex for an MAV. The additional degrees of freedom should be accounted for in the system state that is part of localization and planning for performing efficient autonomous navigation.

1.2.3 *Sensitivity to Uncertainty*

When ground vehicles are subject to noisy measurements or localization with high uncertainty, it is possible to stop completely just to acquire more measurements. Whereas in the context of MAVs, although they are able to hover, this is still a complex control operation owing to their inherent instability of operation. Hence, MAVs are more susceptible to drift because of bad measurements. In turn, this means MAVs need an additional degree of accuracy in their perception/planning modules to ensure good measurements with low uncertainty for safe, efficient navigation.

1.3 MAV Localization

MAVs by themselves are inherently unstable dynamic systems. A key problem in these platforms is the continuous stabilization and control in six degrees of freedom, which can be divided into two steps: attitude control (roll, pitch and yaw angles) and position control (X, Y and Z). Their highly nonlinear and unstable system necessitates a carefully chosen combination of sensor equipment and controller.

Conventionally, an MAV controller is designed as a cascaded structure that contains an inner and an outer loop. The inner loop contains attitude control which runs at a very high frequency, usually at 200-400 Hz to match the dynamics and control frequency of the actuators. The outer loop contains position control that runs at significantly lower rates. As the first step towards control is estimation, attitude control typically utilizes data from inertial measurement units (IMU) that measure angular velocities and linear accelerations of the vehicle.

Although attitude control is capable of making an MAV hover, it is still not possible to estimate and correct for any drift that is caused by an accumulation of errors in measurement. IMUs on most current MAV platforms are made of a combination of micro electro mechanical systems (MEMS) accelerometers and gyroscopes which are susceptible to various errors arising from bias, temperature dependence, repeatability issues etc. that are

hard to model accurately. Hence, it is common practice in MAV platform design to combine IMUs with one or more exteroceptive sensors that can estimate the vehicle's position.

The most common sensor used for this purpose is a GPS receiver. Usually coupled with the IMU, a GPS+IMU system fuses attitude data with position data received from satellites to create a drift-limited, controllable platform. While GPS+IMU combinations are almost ubiquitous on many MAVs today for position and attitude estimation, they do have several drawbacks: GPS signals are not very reliable and may exhibit position error near large structures such as buildings, and they have limited availability in remote areas, while being completely inaccessible indoors. Several alternatives to GPS have been considered for autonomous robots, such as laser rangefinders. Yet, both 2D and 3D laser scanners are not optimal choices for MAV platforms as they are usually large, heavy and have high power requirements that could seriously affect flight time.

The choice of sensing with the most level of success in MAV position estimation has been vision. The earliest implementations of vision based estimation involved setting up external cameras at known positions which can track the MAV - a concept that has now been extended into motion capture systems such as the Vicon,[3] which can perform tracking with millimeter-level accuracy. However, such external systems can only be viable in a research/prototyping setting, and does not help achieve fully autononomous behavior in general environments.

A natural extension to tracking a vehicle using external vision sensors would be to install a vision sensor onboard the vehicle directly: which is capable of estimating its own pose and by extension, that of the vehicle. This allows for making the vehicle fully autonomous, as the sensor is responsible for the vehicle's perception directly. Several types of vision sensors have been evaluated for this purpose, such as monocular cameras, stereo cameras and RGBD cameras. Of these, monocular cameras arise to be the best choice for MAV platforms. Stereo cameras are two monocular cameras separated by a known distance called the baseline: and this baseline is directly proportional to the observable depth from the camera.

As MAVs operate under stringent size restrictions, this limits the baseline of the stereo camera and thereby the viewing distance of the sensor. The other type of vision sensor is an RGBD camera, such as the Microsoft Kinect, and these are active 3D depth estimation setups which employ infrared lasers to perform depth calculations. These sensors usually perform reliably only in indoor areas and are prone to failure when there is high ambient light such as in outdoor areas. To summarize, out of many onboard vision sensor choices, monocular vision offers the best trade-off between size, cost, reliability of information and applicability. Due to this reason, cameras have become one of the most widely installed sensors on many off-the-shelf MAV platforms alongside GPS+IMU sensor combinations. As explained in subsequent chapters, techniques described in this dissertation employ monocular vision based methods to perform localization and path planning.

1.4 Collaborative aerial vehicles

A single autonomous robot in a complex operation can always run the risk of being resource-limited, with no backup in conditions that may lead to its failure. This effect is more prominently seen in the context of MAVs, given their challenges with regards to limitations in sensor hardware, computation as well as flight capability. Recent advances in hardware, sensing and wireless communication have enabled research to focus on the idea of multiple autonomous agents that can collaborate with each other. Collaborative multi-robot systems provide several advantages over single agents, such as robustness due to sharing of data and fusing information, redundancy in case of robot or sensor failure, increased spatial coverage etc. In regard to MAVs, it could be more desirable to employ a team of simpler, smaller, low-power MAVs compared to one high-power, heavy aerial vehicle which could pose problems for flight time. Using multiple MAVs as a group has the potential to boost mission efficiency: by allowing larger spatial coverage, carriage of heavier payloads etc. Multiple MAVs can also be leveraged for task distribution in various schemes. As an example, one or more MAVs with higher computational power can act as leader vehicles, performing the more intensive tasks and commanding the other vehicles as

is required.

Collaboration can be defined as a theme for various tasks performed between the members of a team: these could be sharing and merging information originating from multiple sources, and joint decision-making with the intention of improving existing information and the efficiency of the team. In the first part, the term 'information' means two things: information about the robots themselves: such as state estimates, paths taken for navigating, uncertainties etc. and information about the environment: reconstructed maps, obstacle locations etc. Collaboration and information sharing helps the members manage and distribute tasks in a more efficient way that accounts for their limited resources. On the other hand, joint decision-making ensures that individual actions of the members can benefit the whole team, while helping them determine specific locations or areas to navigate to according to their limitations.

Joint decision-making and information merging especially benefits monocular sensing. One of the major challenges with monocular cameras is the problem of estimating depth: i.e, when viewing an object through a single camera, the image of the object does not afford any information about the size of the object or the distance between the camera and the object. Extended to any general environment, a single camera/image is insufficient to determine the scale of the objects in the environment, which poses a problem for metrically accurate localization and navigation of a vision based vehicle. Binocular vision solves this problem by employing two views; so it can be intuitively seen that multiple vehicles equipped with monocular cameras can collaborate to solve the same problem, thereby acting somewhat as a decoupled stereo camera.

When multiple vehicles function as a group, task distribution can be achieved by enabling collaboration between the vehicles. For example, computationally heavy tasks can be offloaded to a leader MAV with more hardware capacity. Collaboration can also help with localization: if multiple sources of information are available, they can be fused in a consistent manner and the final pose estimates can be made more robust. In the case of

monocular sensing, information fusion can be especially useful: considering that a single camera cannot estimate depth or scale directly by itself, collaboration with other sensors can help resolve these unknown factors.

2. STATE OF THE ART AND CONTRIBUTIONS

The research in the field of autonomous micro aerial vehicles has been very active over the past few years. The relevant research can be mainly classified into broad fields such as localization, control, path planning etc. In the scope of this dissertation, this chapter presents a summary of the state of the art research in localization and mapping using vision sensors and subsequently path planning. Subsequently, this chapter also lists the primary contributions of this dissertation.

2.1 Micro Aerial Vehicle Localization

Traditional sensors used for MAV localization and navigation were GPS receivers combined with onboard IMU sensors. Initial works have examined the efficacy of fusing GPS and IMU data for flight, such as in Abdelkrim et al[4] and Yun et al.[5] To alleviate the problems with GPS receivers on MAVs such as drift and inaccuracy, fusion of information from other sources of data were also investigated, such as a combination of vision, GPS and IMU presented by Templeton et al.[6] Caballero et al[7] present a technique of georeferencing images obtained from the downward facing camera of a UAV and improving pose estimation by propagating this data through a Kalman filter.

In other early works, the potential of vision sensors onboard UAV platforms was recognized although the onboard cameras were not directly used for localization. For instance, camera based navigation was used for helicopters to perform autonomous landing, as presented by Saripalli et al.[8] When vision based localization was first implemented, it was through external camera placement such as in professional motion capture systems, as shown in,[9] or external marker placement such as AR Tags. Many approaches that used motion capture systems for localization were able to capitalize on the extremely precise, high update rate pose estimates to demonstrate impressive results such as aggressive flight, complex and precise maneuvers (Lupashin et al,[10] Hehn and D'Andrea[11]), formation control

for MAV swarms (Kushlayev et al,[12] Ritz et al[13]) etc. As these approaches focused on the control or planning aspects of flight, while assuming reliable and continuous state estimation, their feasibility was limited to controlled environments.

Afterwards, localization using vision sensors as the main exteroceptive sensors started to be of interest. RGBD sensors were one of the initially investigated setups due to the advantage of having direct depth estimates. Stowers et al[14] used a Microsoft Kinect depth sensor on a MAV for altitude estimation. Shen et al[15] presented a fusion of a Kinect depth sensor along with a 2D laser rangefinder for localizing in 2.5D, but creating full 3D maps for the purpose of mapping. Bachrach et al[16] presented localization using an MAV that hosted an onboard Microsoft Kinect, which allowed for RGBD visual odometry. Tomic et al[18] created an approach that combines odometry from a stereo camera as well as a laser rangefinder with onboard computation and data fusion between both estimates. Even more recently, Fang and Scherer[19] showed full 6-DoF localization of MAVs using RGBD sensors. As MAV platforms got smaller, the interest shifted towards monocular cameras, and MAV related research was able to capitalize on the huge amount of progress that was made in the computer vision field on monocular camera based estimation and mapping.

The ubiquity, compactness and relative inexpensiveness of monocular cameras have had a significant influence on the popularity and applicability of the sensor for estimation purposes in the computer vision community for well over a decade. As mentioned earlier, the lack of absolute scale estimation through a single camera has posed several problems in its applicability to MAV localization, due to the necessity of metric positions or velocities in MAV navigation and control. One of the seminal works that presented a monocular localization and mapping method was Parallel Tracking And Mapping (PTAM) framework, originally developed by Klein and Murray,[20] where the scale factor was estimated by a manual calibration sequence during initialization. Soon after, more robust and generally applicable SLAM frameworks were presented through SVO,[21] ORB-SLAM2[22] and LSD-SLAM,[23] among which some were successfully implemented on UAV platforms.

The lack of scale estimation affects all monocular setups, regardless of whether they are performing pose estimation in a perspective-N-point formulation, optical flow approaches or SLAM systems. Methods in the literature have investigated different ways of solving the problem of scale ambiguity. Additional IMU data was used to solve scale ambiguity in the approach presented by Blösch et al.[24] Further advancements in this line of work were shown by Achtelik et al,[17] where vision and IMU data were fused to achieve localization entirely onboard an MAV, using a barometric pressure sensor for visual scale estimation through altitude. Burri et al[25] extend a similar visual-inertial framework to performing mapping, relocalization and planning onboard an MAV in real time. Many other promising monocular visual-inertial systems have been proposed, such as MSCKF,[26] visual-inertial ORB-SLAM2,[27] VINS-MONO[28] etc.

To examine some other ways of solving the depth/scale problem, Bristeau et al.[29] combined a small camera with an ultrasonic rangefinder: while the camera employed an optical flow algorithm to compute planar velocities, the rangefinder would compute altitude, thus removing the scale ambiguity problem. This was implemented commercially in the AR Drone platform. Similarly, Meier et al[30] fused optical flow, ultrasonic altitude data along with altitude estimates from barometric pressure sensors for added reliability, which formed the core of the 'PixHawk' autopilot project. However, these optical flow approaches were only able to compute velocities instead of absolute positions, thus still resulting in occasional drift. Engel et al.[31] provided an enhanced solution where the AR.Drone's operation was combined with PTAM, and scale factors were computed using a maximum likelihood approach. AR.Drone applicability was found to have certain limitations as well, because of the limited range of the ultrasonic sensors and their sensitivity to other sensors and noise. Shen et al[15] use a single camera at a high frame-rate as their primary sensor, but combine this data with another camera of known intrinsic and extrinsic values, that works at a lower frame rate. Pizzolli et al[32] present a very impressive framework that utilizes probabilistic techniques to intelligently estimate depth directly from a monocular video

stream on a per-pixel basis.

2.1.1 Collaborative Localization

Over the past few years, significant work has been done on the theoretical aspects of collaboration for localization purposes. For a generic multi-robot scenario, Martinelli et al[33] present a localization approach that fuses proprioceptive and exteroceptive measurements through an extended Kalman filter, effectively utilizing data fusion. Nerurkar et al[34] present a cooperative localization algorithm that works in a distributed fashion to achieve maximum a-posteriori estimation, but which requires continuous synchronous communication within the robot group. Carrillo-Arce et al[35] present a decentralized cooperative localization algorithm, with the robots only needing robot-to-robot communication when relative measurements are required. Knuth and Barooah[36] propose a distributed algorithm specifically for GPS-denied scenarios, where the robots fuse information from each other continuously and average the relative pose data in order to achieve cooperative estimation that helps all the members. Indelman et al[37] propose a multi robot localization algorithm that uses expectation maximization to take care of the data association problem between the team members, and is also capable of handling unknown starting poses.

Only very recently have there been advances in the realm of collaborative localization using vision and aerial vehicles. For aerial vehicles equipped with monocular cameras, Indelman et al[38] propose a multi-view geometry inspired technique that estimates transformations between the camera views through multi-focal tensors and subsequently localization. Zou and Tan[39] present a collaborative monocular SLAM algorithm with a focus on handling dynamic environments, with multiple vehicles performing individual and relative estimation. Achtelik et al[40] present an approach for two UAVs that each host a monocular camera and an IMU, which is capable of estimating relative poses along with absolute scale, thus acting similar to a stereo camera. Piasco et al[41] treat multiple UAVs with cameras as a distributed stereo system for localization purposes, with a focus on achieving formation control. Forster et al[42] demonstrate a SLAM system that contains a central server

that focuses on features like map merging using information from multiple camera-based vehicles. In this framework, the vehicles do not receive any additional information from the central server or other vehicles in order to benefit from map optimization or pose fusion. Schmuck and Chli[43] present a collaborative monocular SLAM algorithm, also with somewhat of a centralized paradigm: each MAV is responsible for running its own SLAM pipeline and a central server focuses on place recognition, optimization and map fusion.

2.2 Path Planning

Once a robot can localize itself by combining perception and estimation, it assumes the ability to follow certain trajectories and the next step is to compute valid movements from one location to another: this is known as the path planning problem. This problem can be expressed as a partially observable Markov decision process (POMDP) framework, which is known to be computationally intractable.[44] Thus, most of the research in the planning domain has a focus on developing approximate approaches that can scale effectively to the dimensionality and complexity expected in real world problem scenarios.

In particular, sampling based approaches have been found to be highly applicable for combining with SLAM-related frameworks and for real-time implementations onboard vehicles. Sampling based planners discretize the state space using randomized exploration strategies to explore in search of a valid plan, possibly satisfying one or more conditions that define optimality. This approach was adapted into the robot path planning domain with significant success. Influential works in this area include probabilistic roadmap (PRM),[45] rapidly exploring random trees (RRT),[46] which was extended into an asymptotically optimal version RRT* and a graph version RRG by Karaman and Frazzoli.[47] All of these approaches assumed perfect knowledge of the state, deterministic control and a known environment with no consideration of uncertainty. Later, these assumptions were relaxed, and the research community moved its focus towards uncertainty-aware planning approaches, where the robot's belief was no longer perfect. The belief-based counterparts of the previous works were developed as, for example, the belief roadmap (BRM)[48] and the

rapidly exploring random belief trees (RRBT),[49] where predicted uncertainties of future position estimates are considered beforehand. Similar strategies were used to address the planning problem from the context of information gains such as the ones in Levine et al[50] and Hollinger et al.[51] Other belief space planning methods include direct trajectory optimization such as the one developed by Van der Berg et al,[52] which uses a linear-quadratic Gaussian to calculate locally optimal control policies given an initial nominal path, and another method by Agha-Mohammadi et al[53] that attempts to incorporate motion uncertainty in a roadmap framework (FIRM).

While a significant focus was given to the problem of belief-space or uncertainty-aware planning for ground robots, the number of similar approaches that are applied to MAV navigation in full 3D environments is rather low. He et al[54] use a laser-rangefinder system for localization and correctly identify that a direct path between start and goal may not be the best choice for localization, since the measurement range is not considered in a naive plan. Instead, their planner ensures localization by moving the vehicle such that sufficient structure is continuously in view. This was further by Bachrach et al[16] employing belief roadmaps in combination with an RGBD camera as the exteroceptive sensor. Achtelik et al[55] use an RRBT planner to perform uncertainty aware navigation for a single aerial vehicle that is equipped with a downward facing camera. In,[56] a perception aware planning approach is discussed for aerial vehicles, where instead of feature based methods, photometric information of a scene is directly expressed as an information metric in order to achieve perception-aware planning. More recently, Bircher et al[57] presented a receding horizon next best view planner for a stereo camera, with the target application being 3D exploration for an MAV. Falanga et al[58] attempt to unify control and uncertainty-aware planning by formulating the optimization of perception related objectives in a model predictive control framework that works in a receding horizon fashion.

Collaborative path planning for multi-robot systems has been studied for various goals. Some of the earliest research in the idea of cooperative path planning involved extensions

to algorithms such as A*, such as space-time A*.[59] Hennes et al[60] introduced a collision-free navigation method for multi-robot systems in the presence of localization uncertainty. Belkhouche and Jin[61] presented a collaborative path planning approach that utilizes real-time collision detection between the team members, and subsequently modifies positions and orientations of the robots such that collisions can be avoided. Mellinger et al[62] demonstrate an approach capable of planning optimal paths while also enabling formation control for multiple heterogeneous MAVs. The problem of goal assignment and spatial traversal for large teams of MAVs was investigated by Turpin et al.[63]

2.3 Contributions

In the previous sections, the challenges in MAV autonomous localization and navigation were presented, along with the advantages that multi-MAV teams and subsequent collaboration can bring to the pipeline of autonomy. With the aim of addressing some of these challenges, and attempt to answer them from the context of collaboration, this dissertation presents two major frameworks: one for performing collaborative localization and one for collaborative uncertainty-aware path planning. The methods discussed herein are applicable to groups of micro aerial vehicles incorporating a minimal sensor setup, which is only a monocular camera. Monocular cameras offer a good combination of characteristics such as low weight, wealth of information and ubiquity which make them appealing candidates for MAV sensing. This dissertation investigates and proposes two methods: a feature-based collaborative localization method for MAVs utilizing monocular vision, and as an extension to that localization system, a collaborative path planning for a group of MAVs also geared towards feature-based vision. The core contributions are listed in detail below.

1. Development and evaluation of a vision-based collaborative localization (VCL) algorithm. This algorithm uses solely images from a monocular camera for each MAV, and utilizes a structure-from-motion inspired feature based pipeline to achieve localization in a mostly decentralized fashion. As the first step, the MAVs communicate feature

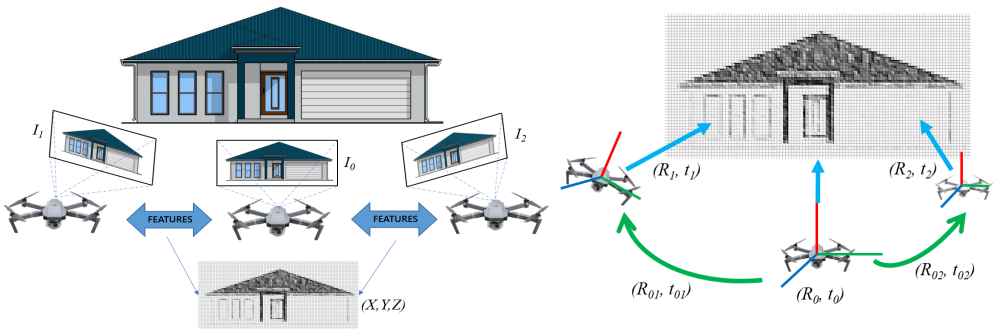


Figure 2.1: The two main phases of vision based collaborative localization: collaborative mapping (left) and subsequent localization through data fusion (right). Reprinted from Vemprala and Saripalli[64]

data between each other to isolate common features and reconstruct a 'map' in 3D. Once a map is available, two types of localization can be performed: intra-MAV localization, which is performed by each vehicle to obtain its own pose estimate; and inter-MAV localization, which involves two or more MAVs to generate a relative pose measurement. All pose estimates are coupled with estimates of measurement uncertainty, which is used to compute pose covariances through a Kalman filter. The VCL framework allows for robust fusion of relative and individual pose estimates in order to maximize accuracy. Figure 2.1 shows a visual representation of the two phases of VCL.

2. Development and evaluation of a vision-based collaborative path planning (VCP) algorithm. This serves as an extension to the VCL algorithm: assuming a group of vehicles is able to use the VCL algorithm to localize, the VCP framework attempts to achieve uncertainty-aware navigation by performing two functions. The first one involves trying to improve the existing 3D maps through a multi-vehicle next best view algorithm, where multiple vehicles collaborate to identify which viewpoints can result in a better map than what is already present, which can subsequently be

Figure reprinted with permission from 'Monocular Vision based Collaborative Localization for Micro Aerial Vehicle Swarms' by Sai Vemprala and Srikanth Saripalli, 2018, Proceedings of the 2018 IEEE International Conference on Unmanned Aircraft Systems, © 2018 IEEE

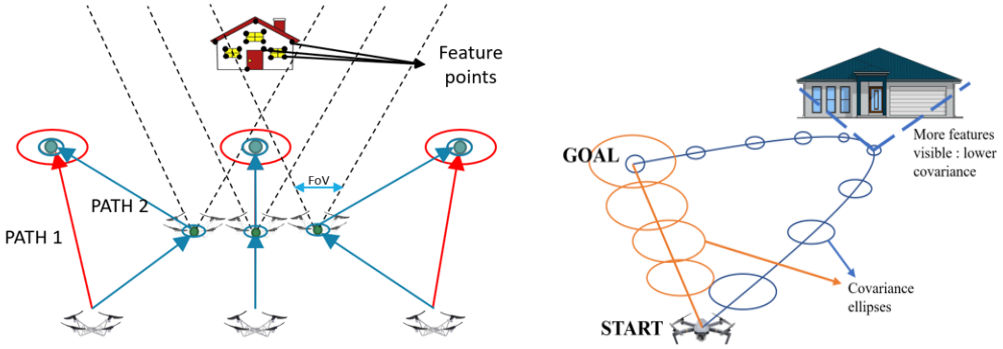


Figure 2.2: The two main phases of vision based collaborative path planning: covariance reduction can be achieved through either improving existing maps (left) or by improving feature tracking on the map (right). Reprinted from Vemprala and Saripalli[66]

shared with all vehicles. Once an improved map is available, the VCP framework executes a sampling-based planner known as Vision-Aware RRT* (VA-RRT*) that runs individually for each vehicle. VA-RRT* plans paths for each MAV in such a way that localization is not compromised, and is improved wherever possible: thus attempting to reduce the vehicle's pose covariance. Figure 2.2 shows a visual representation of the two phases of VCP.

The overall goal of the ideas presented in this dissertation is to provide solutions to groups of MAVs to navigate in environments where collaboration can be beneficial and a high localization accuracy is essential. The VCL framework avoids having to run individual SLAM pipelines on each MAV by treating the mapping function as a distributed task between MAVs. For example, depending on the architecture of the target system, matching and reconstruction can be offloaded to one MAV that has access to better computational modules, thus acting as a leader, and other MAVs with weaker computers can perform only feature tracking and localization. At the same time, the vehicles can interact with each other directly through the relative pose estimation. The problem is not formulated as SLAM, so the focus on mapping is relatively lower: the uncertainty of map points is ignored, and

the map is not part of the system state. While this system can be capable of handling dynamic environments through timely map updates, it still operates under the assumption of sufficient feature overlap over time.

A closely related combination of localization and path planning ensures that collaborative constraints are taken into account sufficiently in different phases of navigation. With the focus on uncertainty-aware planning, it is ensured that the MAVs maintain observability of features while navigating, thus reducing possibilities of drift or crashes. All of these tasks have been achieved solely with data from a vision sensor without the help of any external sensing such as GPS or motion-capture systems, thus making it applicable for GPS-denied scenarios. The techniques presented in this dissertation were validated on a combination of both simulated and real tests.

The tests did not involve real-time implementation of these algorithms, hence the algorithms were implemented on pre-recorded data in an offline fashion. Regardless, the dissertation contains discussion on the feasibility of applying these algorithms real-time through analyses of timing, communication requirements etc. While these algorithms were not implemented in real time and were only tested independently, the target application scenario envisages the VCL and VCP frameworks working together in a loosely coupled fashion. The localization algorithm can be assumed to run continuously, with the planning algorithm attempting to contribute to better localization. Thus, the VCL results such as localization data and uncertainty act as feedback for the VCP algorithm, allowing it to identify better trajectories or viewpoints.

The work in this dissertation is a combination and extension of the author's prior research in the domain of collaborative localization,[64][65] and uncertainty-aware collaborative path planning.[66]

3. VISION BASED COLLABORATIVE LOCALIZATION

3.1 Problem Statement

This chapter presents a solution for estimating the poses of multiple micro aerial vehicles navigating in full 3D space, i.e., utilizing the full six degrees of freedom. Each micro aerial vehicle is defined as a multirotor platform capable of moving in all 3 directions with full roll, pitch and yaw capabilities, thus able to traverse the space of $\mathbb{R}^3 \times SO(3)$, and is equipped with an intrinsically calibrated monocular camera. The cameras are assumed to be facing forward and capable of capturing images of the environment in front with a finite, known field of view. Within the world frame of reference, the position of the camera is considered to be equivalent to the position of the vehicle, as the vehicles are usually small in size. Thus, for k MAVs in a group, the goal is to estimate a full system state $\mathbf{X}$ where each state is a combination of the 3D position of the vehicle and the roll, pitch, yaw angles in a predefined frame of reference.

$$\mathbf{X} = \begin{bmatrix} \mathbf{X_0} & \mathbf{X_1} & ... & \mathbf{X_k} \end{bmatrix} \tag{3.1}$$

$$\mathbf{X_k} = \begin{bmatrix} x_k & y_k & z_k & \phi_k & \theta_k & \psi_k \end{bmatrix} \tag{3.2}$$

The cameras are assumed to adhere to the central projection (pinhole camera) model. Each camera is described by an intrinsics matrix $\mathbf{K}$ (which is known beforehand) and is capable of producing $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ projections for each 3D point it observes through a projection function π. In Appendix A, the camera model and projective geometry are described in detail. Given this information, and a set of images from each camera captured at every instant, the responsibility of the localization algorithm is to estimate the corresponding

Some figures in this chapter are reprinted with permission from 'Monocular Vision based Collaborative Localization for Micro Aerial Vehicle Swarms' by Sai Vemprala and Srikanth Saripalli, 2018, Proceedings of the 2018 IEEE International Conference on Unmanned Aircraft Systems, © 2018 IEEE

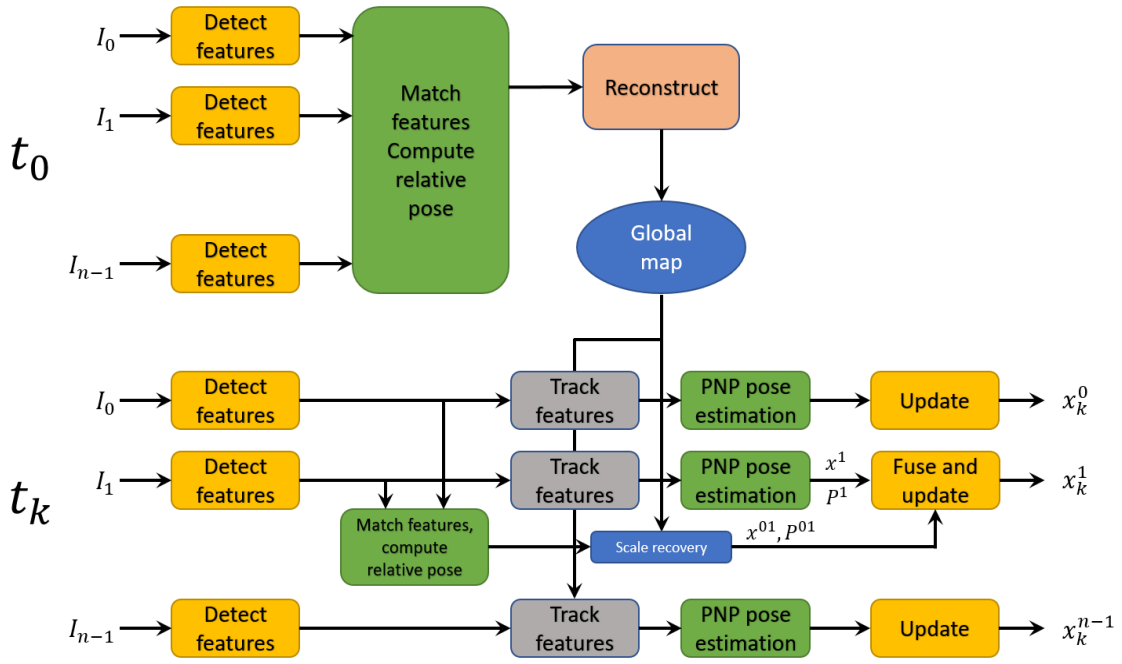


Figure 3.1: Vision based Collaborative Localization (VCL): process flow. Reprinted from Vemprala and Saripalli[64]

6-DoF poses of all MAVs to form the state matrix specified above. As detailed in the following sections, the pose estimation problem is solved in a decentralized way: based on the decomposition of world-to-image and image-to-image transformations, thus achieving both individual pose estimation for each vehicle and relative pose estimation between vehicles when needed.

3.2 General framework and assumptions

The general flow of the VCL algorithm can be seen in figure 3.1. The individual blocks represent various parts of the algorithm, and these parts will be discussed in detail in the following sections.

To facilitate pose estimation, three important assumptions are made in the context of the localization algorithm:

1. All the cameras onboard the vehicles are calibrated, and the intrinsics (and distortion coefficients, if any) are known.

2. The true distance between at least two of the vehicles in the group is known prior to commencement of flight.

3. Communication delays between vehicles are ignored in the current scope of the work. Any information transmitted between vehicles is assumed to be received without delay or loss.

3.3 Feature detection and matching

The pose estimation strategy used by the localization framework is addressed as a feature-based method. Feature-based methods are a subset of vision based methods which attempt to describe an image through a set of features, as opposed to direct methods, which use intensity differences between images as a source of information. A feature, also known as interest point or keypoint; is defined as a point that, when combined with its immediately neighboring pixels, possesses a specific pattern generally associated with one or more distinguishable properties. These properties may include corners, regions, edges and so on. Features represent essential anchor points that can summarize the content of an image and can also help in searching for specific regions between images. Hence, a feature is a location in the image plane with X and Y coordinates, combined with a numerical descriptor that represents the pixel information of the point and its surrounding pixels. The phrase feature detection traditionally refers to the algorithm or technique that detects local features and prepares them to be passed to another processing stage that describes their contents, i.e. a feature descriptor algorithm. In this work, the phrase feature detection is used to broadly mean a combination of the steps of feature extraction as well as description. Figure 3.2 shows an example of how feature detection looks for a sample image.

The literature is rich with a variety of feature extraction and description algorithms. Some of the influential algorithms presented in the last decade or so are SIFT,[67] SURF,[68] FAST[69] and ORB.[70] Out of all these methods, only some are termed as invariant methods, i.e., they are able to identify and locate interest points even under changing image conditions

Figure 3.2: Salient features identified and marked in green for a sample image.

such as scale, illumination, rotation etc. As the focus of the localization algorithm is onboard micro aerial vehicles which exhibit fast movements, transitions between various lighting conditions, invariance is essential for a feature detection algorithm that may be used for MAV camera images. Two methods were chosen, classified by their type of implementation and are listed below:

1. AKAZE: Accelerated KAZE features and M-LDB descriptors: CPU implementation
2. KORAL: Multi-scale FAST features and LATCH descriptors: GPU implementation

3.3.1 A-KAZE features

A-KAZE features were presented by Alcantarilla et al[71] an extension to their previous work that proposed KAZE features. As mentioned before, invariance is an important property of feature detectors, and scale invariance is usually achieved by examining an image at different resolutions, or different scales: also known as the task of constructing a 'scale space'. KAZE features exploit non-linear scale spaces by using non-linear diffusion filtering. This technique makes blurring in images locally adaptive to feature points: thus reducing noise while retaining the boundaries of regions of interest. The KAZE detector works using the scale normalized determinant of the Hessian matrix that is recomputed

at various scale levels. The maxima of these regions of interest are picked up using a moving window approach and are treated as salient features. A-KAZE uses a more efficient algorithm to construct the non-linear scale space, known as Fast Explicit Diffusion (FED). AKAZE features are invariant to scale, rotation, limited affine and have more distinctiveness at varying scales because of nonlinear scale spaces, while being faster than SIFT/SURF detection algorithms.

A-KAZE features are combined with a binary description scheme. Binary descriptors provide a significant advantage over their floating point counterparts (such as the ones used on SIFT/SURF) because of their low memory footprint, thus being applicable for algorithms of a cooperative nature that might require communication of feature data. The descriptor used is known as a modified Local Binary Descriptor (M-LDB) that condenses the description of surrounding pixel data into a 488-bit vector.

3.3.2 KORAL features

Alongside the earlier implementation, this algorithm has also been tested with a GPU implementation of multi-scale FAST features, known as KORAL. The FAST algorithm was originally proposed by Rosten and Drummond,[69] which is a technique for determining corners in a given image by examining a circle of 16 pixels around a candidate pixel. The intensity differences between the pixels are used to classify it as a corner. FAST generally exhibits superior performance to many feature detection algorithms in terms of computation time and resource utilization, thus making it suitable for real-time processing. As a downside, FAST is susceptible to noise and generally fails to detect good features if the image exhibits blur. To counter this, FAST can be extended to an image pyramid of multiple scales: which still provides a considerable speedup over conventional multi-scale detectors.

In KORAL, a candidate image is first divided into 7 scales in a pyramid formation, each scale containing the image in smaller resolution than the parent: and FAST is run on each layer of the pyramid to isolate salient features. As FAST is computing keypoints for each layer, a GPU back-end performs high speed bilinear interpolation to resize the image into

the next layer of the pyramid. Once the keypoints are computed throughout the nonlinear scale space, KORAL uses LATCH description to compute descriptors for each salient feature. LATCH (Learned Arrangements of Three patCH codes) is a recent advancement in feature descriptors presented by Levi and Hassner.[72] While normal feature description methods rely on comparison of individual pixel values between two or more patches, LATCH opts for a triplet approach: where triplets of patches are compared in order to set the binary values of the representation In this algorithm, the CUDA version of LATCH was used,[73] which results in 512-bit descriptor vectors for each feature.

3.3.3 Feature Matching

Once a set of salient features are extracted from all images in consideration, the common points between the images are identified using a step of feature matching. Feature matching involves isolating the set of points that are closest to each other in all the images. Considering only two images for simplicity, each 'match' is a pair of features whose descriptors are closest to each other among all other possible feature combinations. In both the A-KAZE and KORAL methods, the descriptors are binary vectors: and hence, a Hamming distance metric was used to perform matching in a brute-force fashion. Hamming distance between two binary vectors identifies the number of positions where the bits are different: so, a low distance means the descriptors are very similar to each other. Similar to the detector implementations, the matching algorithm for KORAL also runs on CUDA, providing a significant speedup compared to its CPU counterpart.

3.4 Relative pose estimation

This section describes relative pose estimation of two cameras without known 3D poses and by only observing a previous unknown scene. Due to the unknown scene, no 3D/2D correspondences are available: only 2D/2D correspondences between image points can be obtained. The detection and matching step described earlier is responsible for gathering these correspondences, each correspondence containing a combination of keypoint locations

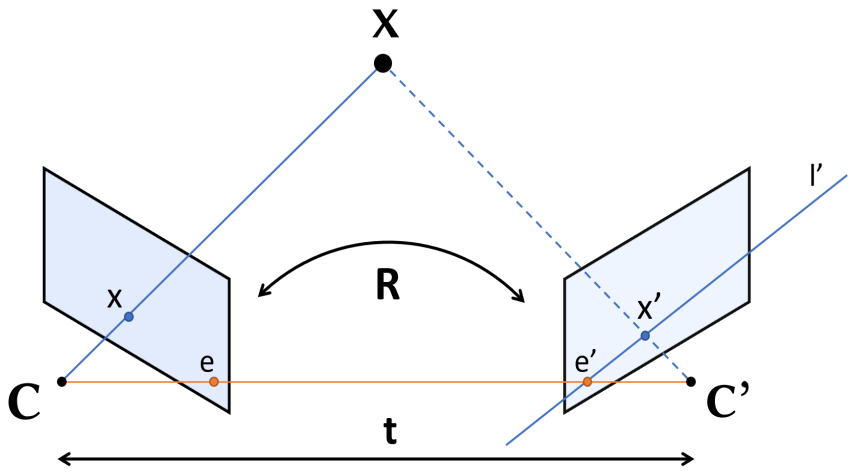


Figure 3.3: Epipolar geometry

(pixel coordinates of the feature) in two or more images. This information is then used to compute both the relative pose between the cameras and a sparse reconstruction of the scene being observed.

A simple case of relative pose estimation is shown in figure 3.3. Assuming two cameras arranged to the left and right are viewing a 3D point X, each of these 3D points would have two projections on both the image planes of the cameras. If such a point common to both the cameras exists, these two views would then be related through the concept of epipolar geometry. The left image contains a point x that originated from the unknown 3D point X, whereas the same 3D point is projected as x' in the right camera. According to the pinhole camera model, X and x can be connected through a line $\mathbf{l}$ that eventually passes through the camera's center. This line, known as the epipolar line, has its own projection $\mathbf{l}'$ in the right camera; as does the center of the left camera. This projected center is known as the epipole (e' in the figure).

The epipolar line projected on the right camera can be described as the line connecting the epipole and the projection of the 3D point on the right camera, thus:

$$\mathbf{l}' = e' \times \mathbf{P}'X = \mathbf{t}' \times (\mathbf{R}'x + \mathbf{t}') = [\mathbf{t}']_{\times}\mathbf{R}'x \tag{3.3}$$

As the epipolar line $\mathbf{l}'$ also contains x', this condition can be expressed as follows, which in turn results in a relationship between the projections of X in both cameras.

$$\hat{x}'^{\top}\hat{\mathbf{l}}' = 0 \implies \hat{x}'^{\top}[\mathbf{t}]_{\times}\mathbf{R}\hat{x} = 0 \tag{3.4}$$

The matrix $\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}$ is called the essential matrix, and it encodes the epipolar geometry between the two views as a combination of the rotation and translation between the two camera centers. It can be constructed given a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ with three degrees of freedom. Although it combines both of those quantities, $\mathbf{E}$ only has five degrees of freedom, and thus is invariant to scaling. Thus, given an essential matrix only, it is impossible to determine the magnitude of $\mathbf{t}$, i.e., when decomposed, only a unit vector is obtained. When the internal camera parameters are known, a finite number of correspondences can be used to estimate the essential matrix. As the epipolar geometry depends on five parameters: 2 for translation, and 3 for the 3D rotation ,a minimum of 5 matches are required between two images to estimate the essential matrix. Specifically, the relative pose estimation algorithm used within the VCL framework uses a solution presented by Nister[74] that uses a five point correspondence minimum between the images to estimate $\mathbf{E}$.

The essential matrix can then be decomposed to obtain the actual rotation and translation parameters. Up to the scale of $\mathbf{t}$, $\mathbf{E}$ can return 4 theoretical solutions, but with only one of them being valid.[75] This validity is determined by what is known as the chirality constraint, i.e, only one of these solutions for the pose ensure that the scene points are actually in front of both the cameras in question. The approach used to compute the valid solution from $\mathbf{E}$ involves a singular value decomposition of the matrix as follows.

$$\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{V}^{\top} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}'_1 \\ \mathbf{v}'_2 \\ \mathbf{v}_3 \end{bmatrix} \tag{3.5}$$

Given this decomposition, it can be shown that the rotation matrix and translation vector assume the following combinations.

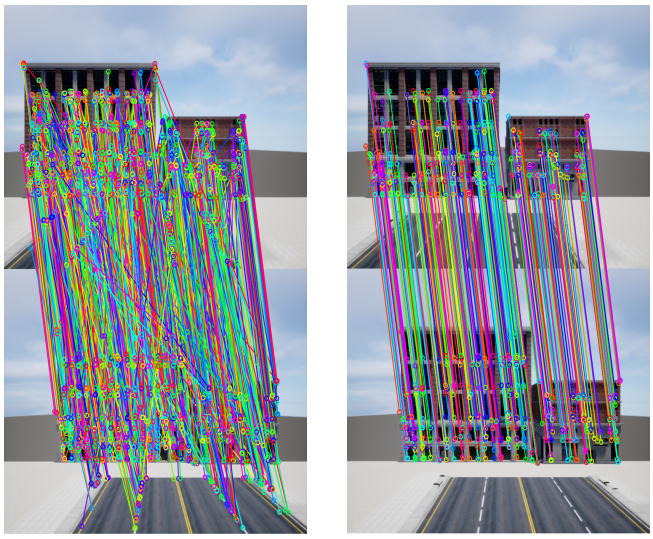
$$\mathbf{R} \in \begin{pmatrix} \mathbf{U}\mathbf{W}\mathbf{V}^\top & \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \end{pmatrix} \text{ where } \mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$\mathbf{t} = \pm\lambda\mathbf{u}_3 \tag{3.7}$$

The correspondences obtained between the images (feature matches) are usually not perfect, i.e., they are a combination of both accurate and inaccurate matches, which can be attributed to texture repetition in the images or similar looking objects distributed over the image. To alleviate this problem, estimation of the essential matrix is usually combined with a random sample consensus (RANSAC) scheme. RANSAC is an algorithm that is used to estimate parameters of a model that the data adheres to, among which some outliers exist that do not respect the model. RANSAC operates under the assumption that there is a consensus among the inliers, whereas the outliers are random. The process usually involves iteratively selecting small sets of the given data, estimating the parameters of the model and evaluating the residual error of the fit using a predefined threshold. It can thus be seen that RANSAC can suffer from the problem of an arbitrary choice, since an accuracy threshold has to be preset, which could be problematic for a situation like VCL where the image characteristics and noise levels could change significantly in the application domain of the RANSAC algorithm.

To avoid the problem of having to pick an accurate threshold, the VCL algorithm uses a modified scheme known as the a-contrario RANSAC (AC-RANSAC). Proposed by Moisan and Stival[76] and demonstrated for computer vision applications in the subsequent work by Moisan et al,[77] AC-RANSAC has three main features:

1. The threshold for inlier/outlier discrimination is adaptive, it does not need to be fixed.



(a) Raw feature matches

(b) AC-RANSAC filtered matches

Figure 3.4: Demonstration of how AC-RANSAC helps with filtering out outliers in feature matches.

2. It gives a decision on the adequacy of the final model: it does not provide a wrong set of parameters if it does not have enough confidence.

3. The procedure to draw a new sample can be amended as soon as one set of parameters is deemed meaningful: the new sample can be drawn among the inliers of the existing model.

This a-contrario approach is capable of choosing the value of parameter T adaptively, based on the noise in the given data, thus enabling robust choice making in terms of the finding the right model, i.e., the essential matrix. Through this choice, the feature matches are filtered to remove outliers, and then the essential matrix is decomposed into rotation and translation. Figure 3.4 demonstrates how AC-RANSAC helps in filtering outlier matches during essential matrix estimation.

3.5 Map building

The relative pose estimation step ensures the transformation between two or more views is known. Next, this information can be used to compute a sparse reconstruction of the

surrounding environment. The well known Hartley-Sturm triangulation method[78] is used to combine the transformations with the known projections of the matched points to obtain the 3D locations of the matched points. This map is then published as a global source of information, accessible from all the MAVs. This map is the seed information from which feature tracking, pose estimation etc. are done. The globally available map data consists of a list of all 3D locations of all the points, as well as the corresponding feature descriptor vector associated with each point. The images responsible for this reconstruction are known as 'keyframes', and the descriptors of the common matches from these images are stored in the map data.

It is useful to recall here that according to assumption 2 in section 3.2, the distance between at least two vehicles is already known prior to initialization. This helps remove the scale ambiguity problem with the very first reconstruction. For a group containing more than two vehicles, all MAVs capture images from their cameras, and a visibility graph of feature overlap is generated in order to isolate all combinations of common features. The pair with the highest number of overlapping features is considered a seed pair and a first reconstruction is attempted with only those two views. Once this reconstruction is generated, other MAVs are incrementally included in this reconstruction based on the features that are observed from their cameras. Finally, all the poses and scene are refined using a sparse bundle adjustment method. Considering the fact that this problem is formulated as solely a localization problem and not SLAM, during/after reconstruction, the uncertainty of the map points is ignored, and the map points do not form part of the system state. It will be discussed in the next sections how the currently accurate metric scale value can be propagated to future reconstructions for relative pose measurements or map updates.

3.6 Intra-MAV localization

The first phase of the VCL algorithm involves relative pose estimation and constructing a sparse representation of the environment the vehicles are facing, which is known as the map. After a map is available, the vehicles are free to fly in full 3D, under the assumption that the

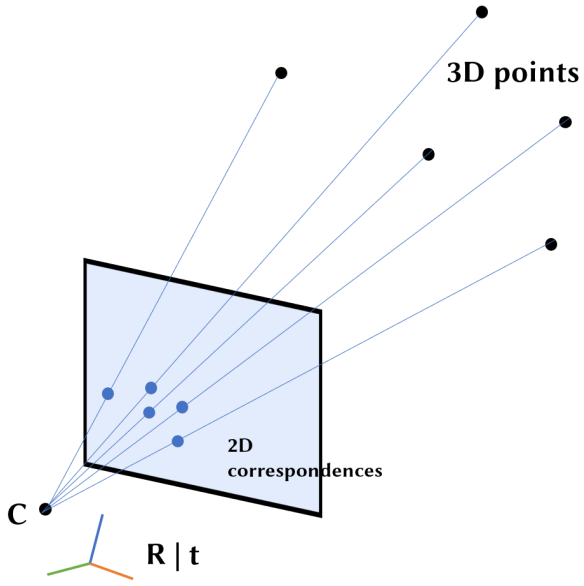


Figure 3.5: Perspective-N-point: The problem of determining rotation and translation of a camera, given the positions of a set of 3D points and their corresponding projections onto an image.

vehicles will continue to capture images and detect salient features in subsequent instants of time. Through this process, the vehicles are made capable of localizing themselves individually, through a process called 'intra-MAV' localization.

The intra-MAV localization method uses a standard image based camera localization technique that attempts to estimate the six degrees of freedom pose of a camera given a 2D image containing projections of known 3D points. Under this framework, every new image is processed through the feature detection step to obtain points of interest, and these points are compared to the features that formed the map, i.e., the features from the keyframes. If a sufficient number of points from the map are still visible in this new image frame, it can be recalled that these points are described by both 2D points on the image plane, as well as 3D points in the real space. The camera pose can then be computed by solving what is commonly known as the 'perspective-N-point' problem on these 2D-3D correspondences (shown in figure 3.5).

Given n 3D points denoted as $\mathbf{X_i} = [\, x_i \; y_i \; z_i \,]$ for $i = 1, 2...n$, and their corresponding 2D projections denoted as $\mathbf{x_i} = [\, u_i \; v_i \; 1 \,]$, the 3D and 2D points are related through projective

geometry as

$$\lambda_i \mathbf{x_i} = \mathbf{R}\mathbf{X_i} + \mathbf{t} \tag{3.8}$$

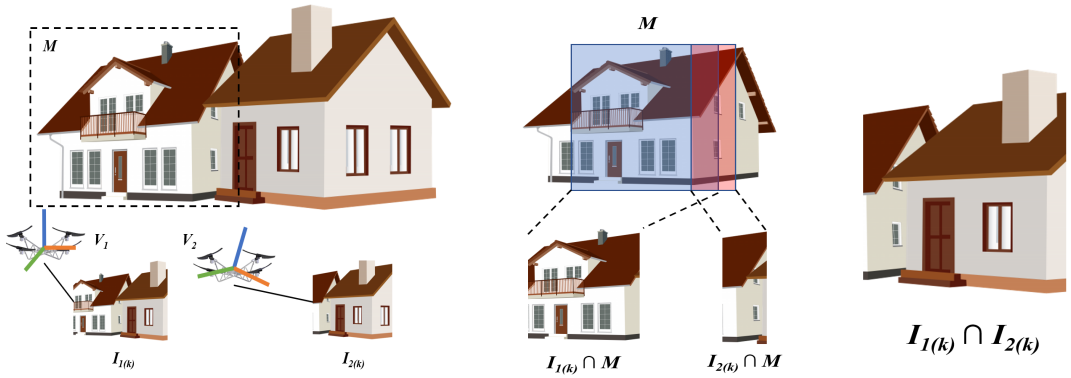
Extending this relationship to a calibrated camera, the equation can be reformulated as

$$\mathbf{x_i} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{X_i} \tag{3.9}$$

Hence, the objective is to determine the values of $\mathbf{R}$ and $\mathbf{t}$. In the current implementation, intra-MAV localization is solved through a combination of a novel perspective-3-point algorithm[79] with another AC-RANSAC scheme, which is then applied to the tracked correspondences between the image and the 3D map, which results in an estimate of the position and orientation. Once this seed pose estimate is computed, it can then be refined further, by attempting to minimize a quantity known as the reprojection error. The reprojection error for an image is the distance between the actual 2D points of the tracked features in the image, and the 2D points that result from reprojecting the 3D points into the image plane from an arbitrary pose θ. π encodes the camera projective transformation of a 3D point $\mathbf{X_i}$ onto the image plane for the computed pose θ, whereas $\mathbf{x_i}$ is the actual observation from the image at that time step. Hence, the aim of this minimization is to obtain the pose θ that results in the least reprojection error, and thus is closest to the real pose of the MAV. This optimization can be expressed as follows.

$$\theta^* = \arg\min_{\theta} \sum_i \|\mathbf{x_i} - \pi(\mathbf{X_i}, \theta)\| \tag{3.10}$$

Through this step, it is also possible to obtain the solution quality, typically expressed as a covariance matrix which is useful for knowing how uncertain the pose estimation is. More details about the uncertainty estimation are discussed in section 3.8.



(a) Example where vehicle V_2 has bad feature tracking.

(b) V_1 has sufficient overlap with map, whereas V_2 does not.

(c) V_1 and V_2 have their own separate overlap.

Figure 3.6: Example scenario where intra-MAV localization could fail, but inter-MAV localization could help.

3.7 Inter-MAV localization

The success and the accuracy of intra-MAV localization depends on the quality and quantity of the features that are successfully tracked in a current image of a certain MAV from the features that comprise the global map. Although the P3P algorithm requires a minimum of only 3 tracked 2D-3D correspondences to constrain a camera pose in full 3D, due to the presence of outliers and noise, in practice, a much higher number of tracked features is required for an accurate pose. If an insufficient number of features are tracked by a vehicle, the error in pose estimation would increase drastically, which could then lead to drift. In a group of multiple vehicles, the existence of other vehicles with their own cameras can be capitalized upon to assist with localization. Hence, an alternative way of performing localization within the VCL framework is built upon relative pose estimation, which allows one MAV to compute the rotation and translation of another MAV in the group, given sufficient feature overlap between the two cameras. This way of performing localization is known as 'inter-MAV localization'.

Figure 3.6 show a possible scenario where inter-MAV localization can be helpful. At time instant k, vehicle V_1 has a good amount of tracked features from the map M (blue

shaded region in 3.6.b), whereas vehicle V_2 does not (red region in 3.6.c). On the other hand, there is sufficient feature overlap between V_1 and V_2 independent of the map; with a non-zero overlap with the map. In a situation like this, inter-MAV localization lets V_1 estimate the metric pose of V_2, which can then eventually be fused with V_2's own estimate.

3.7.1 Relative pose estimation

The process for performing relative pose estimation initially follows the same approach as the one described in section 3.4. The features visible from the images captured by two MAVs V_i and V_j are isolated and matched, and the essential matrix is used to compute a relative rotation and translation. This relative pose, as discussed previously, is not sufficient for a metric pose computation due to the existence of an arbitrary scale factor. It is essential to recover the scale factor before communicating this pose to V_j. Isolating the matched points between V_i and V_j and combining them with the relative pose also results in a local reconstruction of the features common to V_i and V_j: scaled by the same arbitrary factor. This 'local map' is used to help with recovering the true scale factor.

3.7.2 Scale factor estimation

The computation of a relative pose between vehicles V_i and V_j provides access to another local map of features, the ones that are only common to V_i and V_j. This local map, another set of 3D points with an arbitrary scale, can be referred to as $\mathbf{M}'$. If an assumption is made that both V_i and V_j were able to localize using intra-MAV localization, albeit with varying degrees of accuracy, it can be stated that both V_i and V_j have a non-zero overlap with the existing map. Hence, it follows that there is a non-zero overlap of features between $\mathbf{M}'$ and $\mathbf{M}$.

In order to compute the right scale factor λ for this reconstruction (one that matches the true scale of the global map), the frame of reference for the local map has to be considered. Within this local frame, for two MAVs, the 'host' MAV can be considered to be at the origin $[\mathbf{I}|\mathbf{0}]$ and the 'target' MAV at $[\mathbf{R}|\mathbf{t}]$, where $\mathbf{R}$ and $\mathbf{t}$ are the estimated relative rotation

and translation between the host and client. If any two pairs of common features can be identified between the local and the global map, considering that their true 3D coordinates are already known from the global map, the ratio of the length of a line connecting the local coordinates to the length of one connecting the global coordinates is the true scale factor for the new map. It can be recalled from section 3.5 that (at least the first version of) the global map is considered to be metrically accurate. Once this scale factor is known from the ratios, the relative pose is scaled to its right value, and the reprojection error is minimized to obtain a better estimate.

It is important to note here that while comparing the inter-MAV local map and the global map, any wrong matches between the two sets of points can affect the estimation of the scale factor greatly, because the ratio could be computed between completely different points in the two maps. Hence, the VCL algorithm uses guided matching to ensure accuracy of matching between the two point sets.

3.7.3 Guided Matching

The goal of this step is to perform completely accurate matching between two maps: one of them, the (usually larger) global map, and another, a temporary map obtained by the matches between V_i and V_j. Now, it can be recalled that the host MAV, which is responsible for generating relative poses has an acceptable degree of confidence in its own pose, which means that both the global map and local map are generated from confident poses. According to section 3.5, the map is a combination of 3D locations and descriptors: so if the descriptors comprising the global and local maps are treated as two separate images, the transformation between these two views is already known. Unlike the initial step of relative pose estimation, where matching was performed to isolate the essential matrix and compute unknown poses, here, the process is considered in its inverse: the transformation is already known, and it is possible to 'guide' the matching towards inliers according to this known transformation.

It can be assumed that the features belonging to the global map, or at least, a subset of

them: are from an imaginary camera that resides at pose $[\mathbf{I}|\mathbf{0}]$, and the features from the local map are from a camera that resides at the current pose of the host MAV, say, $[\mathbf{R}'|\mathbf{t}']$. Assuming finite overlap between these maps, there exists a relationship between these two cameras, and therefore, the maps, which can be described using the fundamental matrix as

$$\mathbf{F} = \mathbf{K_2}^{-\top}\mathbf{R}'\mathbf{K_1}(\mathbf{K_1}\mathbf{R}'^{\top}\mathbf{t}')_{\times} \tag{3.11}$$

Given that $\mathbf{R}'$ and $\mathbf{t}'$ are known, it is trivial to compute the fundamental matrix according to equation 3.11. Computing this fundamental matrix allows the implementation of 'guided' matching: i.e., any matches that are not adherent to the proper epipolar geometry, as represented in equation 3.12 can be considered outliers and discarded.

$$\mathbf{x_2}^{\top}\mathbf{F}\mathbf{x_1} = 0 \tag{3.12}$$

Figure 3.7 shows an example of guided matching and how it helps determine outliers. The matches computed by the algorithm between two maps are drawn as colored lines, with the radius of the circles at the endpoints indicating the corresponding epipolar error. The bigger circles correspond to a higher epipolar geometric error, and therefore are matches that are far from satisfying the condition in equation 3.12. In practice, no match corresponds to exactly zero epipolar error, so a pixel based threshold on the error helps discard all matches that exhibit a high error. Inliers are shown as thicker circles that correspond to small radii and thus correspond to low error.

Intra-MAV estimation usually suffers in accuracy when there are not enough features to be tracked from the original map. In such cases, inter-MAV estimation can be helpful as it utilizes common features between the MAVs at that instant and does not require multiple observations over time. Scale recovery in the inter-MAV estimation step requires a minimum of only two pairs of accurate matches between the local and global map, as opposed to intra-MAV estimation, which requires a significantly higher number of tracked features for

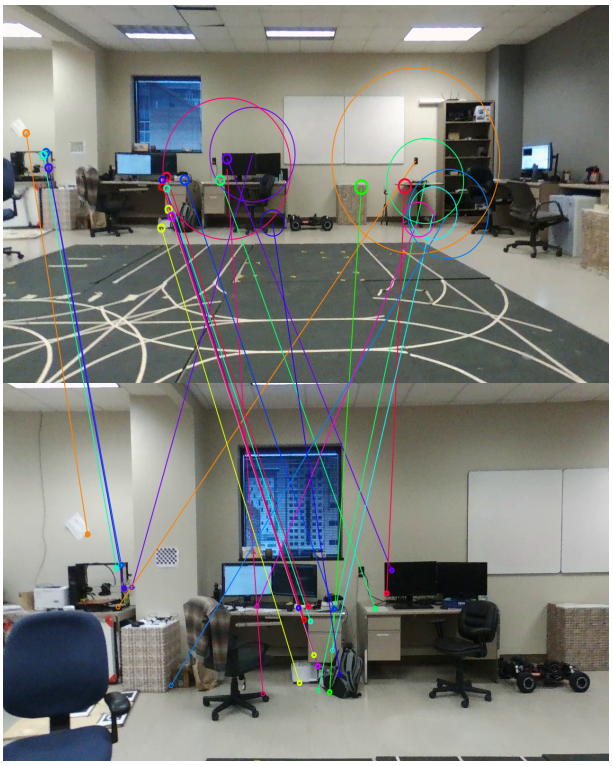


Figure 3.7: Guided matching to remove outliers: false matches are seen to have a high epipolar error, corresponding circles seen to have a much larger radius than those of the inlier matches.

better accuracy.

3.8 Uncertainty estimation

One of the critical parts of localization is to estimate not only the position and orientation of a vehicle, but also estimate the uncertainty of the estimated pose. Since the errors of the sensory system, i.e., the camera, as well as noise from the estimation algorithm affect the final pose, it is important to try and encode camera-related errors in the pose uncertainty. In practice, this uncertainty is described as a covariance matrix.

In both the inter and intra-MAV estimation steps, the final refinement of the pose is performed through a non-linear least squares method where the algorithm attempts to minimize the reprojection error. The reprojection error is a geometric distance, and hence a non-linear function. For a projective transformation π, the cost function that is being

optimized, i.e, sum of all residuals: can be written for m features as

$$f(\theta) = \frac{1}{2} \|\mathbf{r}(\theta)\|^2 \tag{3.13}$$
$$\mathbf{r} = \begin{bmatrix} r_1 & r_2 & \dots & r_m \end{bmatrix} \text{ where } r_i(\theta) = x_i - \pi(X_i, \theta)$$

The Jacobian and the Hessian of this function can be evaluated to be:

$$\begin{aligned} \nabla f(\theta) &= \sum_i r_i(\theta) \nabla r_i(\theta) \\ &= \mathbf{J}(\theta)^\top \mathbf{r}(\theta) \qquad (3.14) \\ \nabla^2 f(\theta) &= \sum_i \nabla r_i(\theta) \nabla r_i(\theta)^\top + \sum_i \nabla^2 r_i(\theta) \\ &\approx \mathbf{J}(\theta)^\top \mathbf{J}(\theta) \qquad (3.15) \end{aligned}$$

When a solution is close to a local minimum, the effect of the second order terms in equation 3.15 is minimized, and hence they can be ignored in terms of their contribution towards the objective. Once the second order terms are ignored, it can be seen that the outer product of this Jacobian matrix at the final optimum with itself is an approximation of the Hessian matrix of the solution. For a non-linear multidimensional least squares error near the optimum, the inverse of this Hessian matrix is an approximation of the covariance matrix of the reprojection errors[80] (see appendix B for a simpler treatment). Hence, the approximate covariance of the solution can be expressed as

$$\mathbf{\Sigma} = (\mathbf{J}^\top \mathbf{J})^{-1} \tag{3.16}$$

Here, it has to be noted that $\mathbf{\Sigma}$ in equation 3.16 does not necessarily translate into an uncertainty in the position/orientation values directly: it is only able to express the quality of the solution and the possible uncertainty around the local surface at the point of convergence. In certain cases, where the solution appears to be a local minimum, the

estimated covariance could still be low while the pose estimate is not true to the actual value. So in order to express the pose uncertainty more accurately, this covariance is inflated further with the reprojection error obtained for that pose estimate, and then used as an estimate of the measurement noise covariance in the update step of the recursive estimator.

$$\mathbf{R} = (\mathbf{J}^\top \mathbf{J})^{-1} * \epsilon_{reproj} \tag{3.17}$$

3.9 Kalman Filter and Outlier rejection

As a final step, the raw measurements obtained during the course of the VCL framework are propagated through a Kalman filter framework. The VCL scheme being a purely vision based localization system without augmentation from other sensors like IMUs, the cameras are essentially considered to be replacements for the vehicles in terms of poses. Due to this reason, the primary function of the Kalman filter in the VCL framework is for smoothing and outlier rejection. Still, this filter can be modified to incorporate a more complex vehicle model, or include IMU measurements as an extension to this work.

Each MAV is responsible for running its own internal recursive estimation scheme through the Kalman filter. At every instant an image is received, it is expected that the MAV computes an intra-MAV pose estimate for itself. Once computed, the obtained measurement is used to correct the state and covariance of that particular MAV. If a measurement obtained at time step k is denoted as $\mathbf{z}_k^i$,

$$\mathbf{z}_k^i = \mathbf{h}(\mathbf{x}_{k-1}^i, \mathbf{x}_k^i) + \mathbf{n_k^i} \tag{3.18}$$

The measurement is then used to correct the predicted pose at time step k, where the

measurement noise covariance is inflated as discussed in section 3.8.

$$\begin{aligned} \mathbf{P}^i_{k|k-1} &= \mathbf{A}^i_k \mathbf{P}^i_{k-1|k-1} \mathbf{A}^{i^\top}_k + \mathbf{Q}^i_k \\ \mathbf{S}^i_k &= \mathbf{H}^i_k \mathbf{P}^i_{k|k-1} \mathbf{H}^{i^\top}_k + \mathbf{R}^i_k \\ \mathbf{P}^i_{k|k} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_{k|k-1} \end{aligned} \tag{3.19}$$

Before using the obtained pose value in the measurement update, it is beneficial to determine the likelihood of the measurement being an outlier. One standard way of performing this check is through what is known as a Chi-squared gating test. The VCL algorithm uses this test to detect and reject obvious mismatches or very noisy observations. Every time a new measurement for the MAV pose is available, the Mahalanobis distance is computed between the expected measurement and the actual measurement as

$$\gamma_k = (\mathbf{z}^i_k - \hat{\mathbf{z}}^i_k)^\top \mathbf{S}^{-1} (\mathbf{z}^i_k - \hat{\mathbf{z}}^i_k) \tag{3.20}$$

The rank of $\mathbf{S}$ determines the system's degrees of freedom, which for a full 6 DoF pose, would be 6. Assuming the process/measurement noises are Gaussian distributed (which is usually the case), γ_k should be Chi-square distributed with 6 degrees of freedom. It is possible to decide upon a certain probabilistic threshold value: which, when exceeded, makes a candidate measurement an outlier. Hence, if γ_k exceeds the α-quantile of the Chi-squared distribution, the measurement can be treated as an outlier and discarded[81,82] .

In the context of inter-MAV estimation, the responsibility of computing both the state and the covariance of the client MAV is taken up by the host MAV. Once a relative measurement between the host and client, denoted as $\mathbf{z}^{i,j}_k$ is available, host MAV V_i computes its own estimate of the state of MAV V_j using the relative measurement as follows.

$$\mathbf{x}^{j'}_k = \mathbf{x}^i_k + \mathbf{M}^{i,j}_k \mathbf{z}^{i,j}_k \tag{3.21}$$

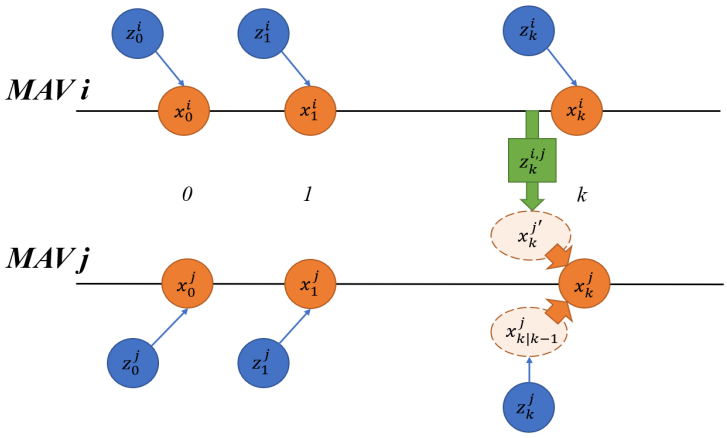


Figure 3.8: Fusion of inter-MAV and intra-MAV estimates: a pictorial representation. Reprinted from Vemprala and Saripalli[64]

While V_i attempts to compute the uncertainty of V_j, this is in combination with V_i's own uncertainty. Consequently, the covariance matrix that V_i has estimated for itself is propagated into any other relative measurements attempted by V_i. When V_i computes a relative measurement to V_j at time instant k with measurement noise covariance $\mathbf{R}_k^{ij}$, the corresponding covariance for V_j: $\mathbf{P}_k^{j'}$ can be calculated as:

$$\mathbf{P}_k^{j'} = \mathbf{H}_k^{j'} \mathbf{P}_k^i \mathbf{H}_k^{j'^\top} + \mathbf{R}_k^{ij} \tag{3.22}$$

3.10 Data fusion

One of the strengths of a collaborative localization routine is the ability to fuse estimated data from multiple sources, thus attempting to result in more robust pose estimates. In the case of the VCL algorithm, it can be intuitively noted that while one vehicle has its own intra-MAV estimate, it could also receive possibly multiple inter-MAV estimates computed by the other vehicles in the group. For simplicity, as before, if two vehicles are considered to form the group, the pose computed by V_i for V_j, say $\mathbf{x}^{ij}$ can be fused with the onboard estimate of MAV V_j itself, $\mathbf{x}^j$, to jointly result in a more robust estimate for V_j (see figure 3.8).

A conventional way of data fusion is to include multiple sources in the update step of a

Kalman filter. The VCL case, however, presents a particular problem: while $\mathbf{x}^{ij}$ and $\mathbf{x}^{j}$ arise from different vehicles, the vehicles have a common source of information: which is the global map. At the same time, both of those vehicles may have exchanged information or pose data in the past either directly or indirectly, (a possibility that is not tracked) which also makes their estimates correlated. Owing to the relative simplicity of the filtering framework so far, these cross-correlation terms are not tracked, and hence the correlation between these two measurements is unknown. Disregarding the cross-correlation entirely can make the conventional Kalman filter update step result in inconsistent and erroneous estimates, resulting in overly confident behavior of the filter in the future. Hence, the VCL algorithm incorporates a fusion approach inspired by the one presented by Carrillo-Arce et al[35] for multi-robot teams, to fuse these estimates using covariance intersection.

Covariance intersection (CI), first presented in the seminal paper by Julier and Uhlmann,[83] is an elegant solution for the fusion of estimates with unknown correlations. The fundamentals of the CI algorithm are rooted in Gaussian intersection (see figure 3.9 for a visual depiction) and its objective is to obtain a 'consistent' estimate of a covariance matrix when two or more estimates are present, which constrains the estimated covariance with an upper bound. That is, for a random variable x with mean $\bar{x}$, an estimate of the mean can be represented as $\hat{x}$ with covariance $\mathbf{P}_x$. The estimation error would be $\widetilde{x} = \hat{x} - \bar{x}$ with its associated covariance $\widetilde{P_x} = E\{\widetilde{x}\widetilde{x}^\top\}$. The state-covariance pair for the estimate is said to be consistent iff

$$\mathbf{P_x} - \widetilde{\mathbf{P_x}} \geq 0 \tag{3.23}$$

The CI algorithm expresses the covariance of the fused estimate as a combination of the individually estimated covariances. Depending on the confidence in each estimate or a desired final statistic, each individual covariance can be weighted by a scalar value. In the case of the VCL framework, each estimate is already described by its own confidence coming either from onboard the same vehicle requiring fusion, or the host vehicle that generated a pose for the client. At time instant k, assume that the individual estimate of V_j

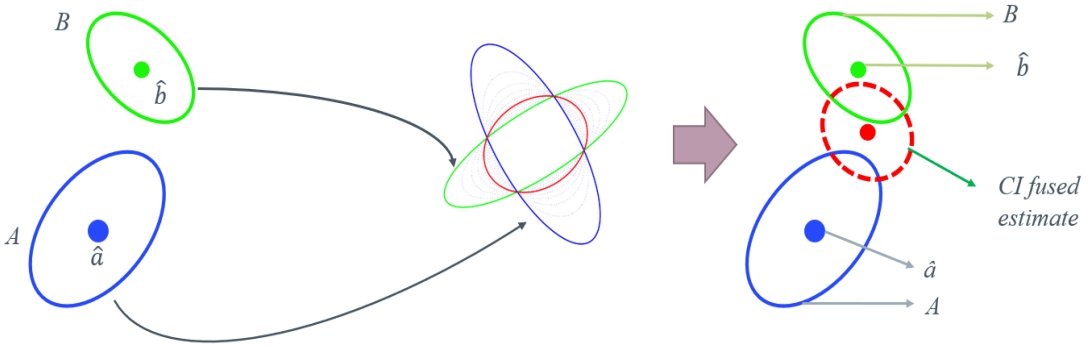


Figure 3.9: The concept of covariance intersection.

is a state-covariance pair with state $\hat{\mathbf{x}}_k^j$ and covariance $\hat{\mathbf{P}}_k^j$. For the same time instant, V_i computes another state-covariance pair for the pose of V_j, represented as $\hat{\mathbf{x}}_k^{ij}$ and $\hat{\mathbf{P}}_k^{ij}$. Then the CI algorithm is used to fuse these two estimates as below.

$$\mathbf{P}_k^j = \left[\omega(\mathbf{P}_k^j)^{-1} + (1-\omega)(\mathbf{P}_k^{ij})^{-1}\right]^{-1} \tag{3.24}$$

$$\mathbf{x}_k^j = \mathbf{P}_k^j\left[\omega(\mathbf{P}_k^j)^{-1}\mathbf{x}_k^j + (1-\omega)(\mathbf{P}_k^{ij})^{-1}\mathbf{x}_k^{ij}\right] \tag{3.25}$$

Where ω is a parameter that is computed such that the trace of the combination of the two covariance matrices is minimized: this can be expressed as in equation 3.26.

$$\arg\min_{\omega} Tr\left[\omega(\mathbf{P}_k^j)^{-1} + (1-\omega)(\mathbf{P}_k^{ij})^{-1}\right]^{-1} \tag{3.26}$$

Another elegant property of the CI algorithm is that, while being derived from a geometric viewpoint, it can also be expressed as a matrix- and scalar-weighted optimization problem. This property allows the CI algorithm to be extended to the fusion of higher dimensional state vectors and thus, an arbitrary number of estimates. Consequently, in the VCL framework, fusion can be performed between more than two sources of data, where for k sources, the covariance matrices are weighted by an array of k weighting factors

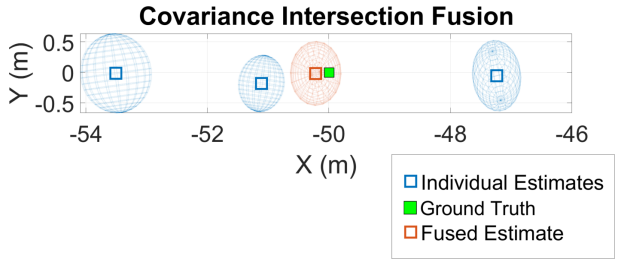


Figure 3.10: Example of covariance intersection fusion of data from three sources.

$\omega_1, \omega_2, ..., \omega_k$ such that

$$\omega_1 + \omega_2 + ... + \omega_k = 1 \tag{3.27}$$

For k sources of data, equation 3.26 can be extended into a multi-variable minimization problem of finding a set of weights that minimize the weighted sum of the traces of all covariance matrices involved. Figure 3.10 is a visual depiction of covariance intersection fusion for data from 3 sources, from an experiment where 3 vehicles were responsible for estimating the position of a fourth on the X axis. At that particular time step, none of the estimates are particularly close to the ground truth, but the level of confidence exhibited by the closest estimate is higher compared to the others; which results in the covariance intersection algorithm computing the right combination of weights to result in a fairly accurate fused estimate.

3.11 Map updates

While the formulation and focus of the VCL algorithm is mainly on localization, mapping is an essential part of the process. In certain situations that could be part of the application, all the MAVs may have to move away from an initial map, which would necessitate an update of the global map in order to maintain localization. While inter-MAV localization is able to assist with the case of specific MAVs leaving the area of the mapped scene, all the vehicles navigating to a different area would require a new map. Hence, the same principles that allow for inter-MAV localization, i.e, relative pose estimation and scale recovery, can be used for building a new global map. This process can be invoked when the tracked

feature count falls under a certain threshold for all the vehicles in the group or any other condition that is deemed appropriate. In a real time scenario, the map update operation would require more communication, because this update would have to be propagated to all the vehicles.

3.12 Implementation and Results

The collaborative localization framework has been written mainly in C++. Various open-source libraries available through OpenCV[84] and OpenMVG[85] were utilized for implementing feature detection, matching, AC-RANSAC and PNP pose estimation. KORAL was used as a separate open source library with CUDA support for the GPU-based algorithms. The Ceres solver[86] was used to refine reconstructions and estimated poses, as well for estimating covariances of the solutions. The refinement of pose estimates through reprojection error minimization was treated as a non-linear least squares problem and solved through an efficient sparse Schur complement method. dlib[87] was used to perform optimization for the covariance intersection.

The collaborative localization algorithm has been tested on datasets obtained from both simulated and real flight tests in an offline manner, i.e., on pre-recorded data. Nevertheless, this section contains some discussion on the real-time feasibility of this algorithm.

For the simulations, Microsoft AirSim was used as the simulation platform. AirSim[88] is a recently developed UAV simulator built as a plugin for Unreal Engine, which is a popular AAA videogame engine with the capability to render high resolution textures, realistic scenes, soft shadows, extensive post-processing etc. in order to bring simulated environments visually close to real life. As the VCL technique is heavily dependent on computer vision, using AirSim enabled testing in high fidelity environments that are close to real life scenarios. Each MAV simulated within AirSim had a forward facing monocular camera, and onboard images were captured at approximately 5 Hz with a resolution of 640×480. A custom urban environment was created in Unreal Engine for the purpose of testing. The images from the onboard cameras and ground truth from the simulator were

Algorithm 1 VCL algorithms: sample for two MAVs

procedure BUILDMAP(x_i, I_1, I_2)
  $i_1, i_2 \leftarrow detectFeatures(I_1, I_2)$
  $\bar{i_1}, \bar{i_2} \leftarrow matchFeatures(i_1, i_2)$
  $\mathbf{E} \leftarrow$ ACRANSAC($\bar{i_1}, \bar{i_2}, \mathbf{K_1}, \mathbf{K_2}$)
  $\mathbf{R}, \mathbf{t} \leftarrow$ SVD($\mathbf{E}$)
  $\mathbf{M} \leftarrow reconstruct(\bar{i_1}, \bar{i_2}, [I|0], [\mathbf{R}, \mathbf{t}])$ ▷ M: Global map
end procedure

procedure LOCALIZEINTRAMAV(I_k, $\mathbf{K_k}$, $\mathbf{M}$)
  $i_k \leftarrow$ detectFeatures(I_k)
  $\bar{i_1} \leftarrow$ trackFeatures($i_k, \mathbf{M}$)
  $\mathbf{R}, \mathbf{t} \leftarrow$ PNP($\bar{p_k}, \mathbf{M}, \mathbf{K_1}$)
  $\mathbf{z_k}, \mathbf{R_k} \leftarrow$ refinePose($\mathbf{R}, \mathbf{t}, \mathbf{M}$)
  $\mathbf{x}_k, \mathbf{P}_k \leftarrow$ updateState(z_k, R_k)
  return $\mathbf{x}_k^j, \mathbf{P}_k^j$
end procedure

procedure LOCALIZEINTERMAV(x_i, I_i, I_j)
  $i_i, i_j \leftarrow$ detectFeatures(I_i, I_j)
  $\bar{i_i}, \bar{i_j} \leftarrow$ matchFeatures(i_i, i_j)
  $\mathbf{E} \leftarrow$ ACRANSAC($\bar{i_1}, \bar{i_2}, \mathbf{K_i}, \mathbf{K_j}$)
  $\mathbf{R}, \mathbf{t} \leftarrow$ SVD($\mathbf{E}$)
  $\mathbf{M}' \leftarrow$ reconstruct($\bar{i_i}, \bar{i_j}, [I|0], [\mathbf{R}, \mathbf{t}]$) ▷ M': Local map
  $m_{map} \leftarrow$ matchFeatures(O', O) ▷ M:= Global map
  $\lambda \leftarrow$ recoverScale(m_{map})
  $[R, t] \leftarrow [R, t] * \lambda$
  $\mathbf{z}_k^{i,j} = [\mathbf{R}, \mathbf{x_i} + \mathbf{t}]$
  $\mathbf{z}_k^{i,j}, \mathbf{R}_k^{i,j} \leftarrow$ refinePose($R, t, \mathbf{M}$)
  $\mathbf{x}_k^{j'}, \mathbf{P}_k^{j'} \leftarrow eqn(7), (8)$
  return $\mathbf{x}_k^{j'}, \mathbf{P}_k^{j'}$
end procedure

procedure FUSEINTERINTRAPOSES($(\mathbf{x}_k^j, \mathbf{P}_k^j), (\mathbf{x}_k^{j'}, \mathbf{P}_k^{j'})$)
  $\mathbf{P_A} \leftarrow \mathbf{P}_k^j$
  $\mathbf{P_B} \leftarrow \mathbf{P}_k^{j'}$
  $\omega \leftarrow \arg\min_\omega Tr(\omega {\mathbf{P_A}}^{-1} + (1-\omega){\mathbf{P_B}}^{-1})$
  $\mathbf{P}_k^{j*} \leftarrow (\omega {\mathbf{P_A}}^{-1} + (1-\omega){\mathbf{P_B}}^{-1})^{-1}$
  $\mathbf{x}_k^{j*} \leftarrow \mathbf{P}_k^{j*}(\omega {\mathbf{P_A}}^{-1}\mathbf{x}_k^j + (1-\omega){\mathbf{P_B}}^{-1}\mathbf{x}_k^{j'})$
end procedure

(a) Simulated city environment in Unreal Engine/AirSim

(b) Parrot Bebop 2 quadrotors used in real flight experiments

Figure 3.11: Implementation details

recorded and the VCL algorithm was tested offline on this data, while being compared to ground truth.

For the real life tests, two Parrot Bebop 2 quadrotors were used in both outdoor and indoor scenarios. Videos from the forward facing monocular cameras were recorded onboard the vehicles at a 1280×720 resolution at approximately 15 Hz, and then processed offline. The GPS/IMU data was also recorded for the outdoor flights for comparison purposes. Pictures of the simulation environment as well as the real vehicles can be seen in figure 3.11.

3.12.1 Intra-MAV localization

3.12.1.1 Simulation

In this experiment, three MAVs were initialized in the urban simulation environment, which were then commanded to take off and fly in square-like trajectories at different altitudes, while onboard camera images were recorded. Within this dataset, the algorithm was first asked to build a global map using images from the three vehicles and intra-MAV localization was tested for images corresponding to each vehicle. Figure 3.12 shows the three trajectories of the vehicles, and compared to the ground truth, table 3.1 shows the mean squared errors of the VCL estimates for the three vehicles. It can be seen from the values in table 3.1 that intra-MAV localization exhibits a good amount of accuracy.

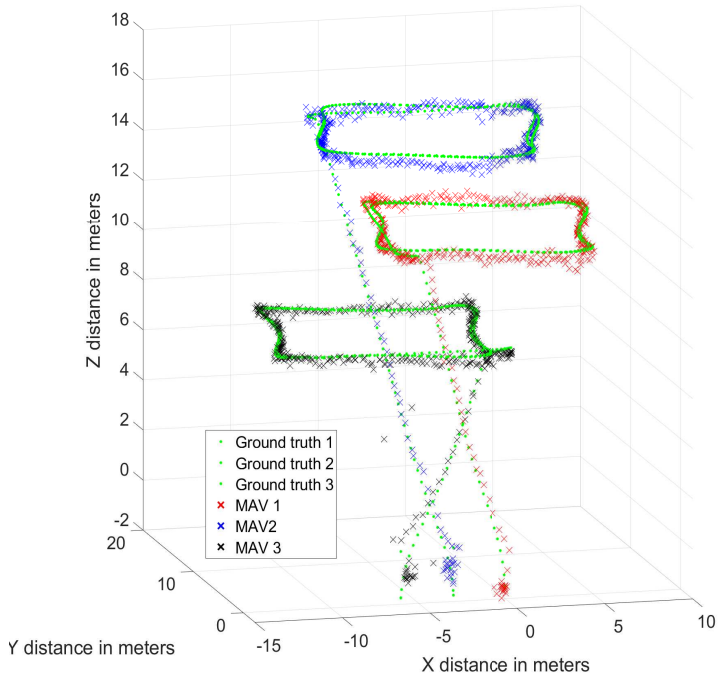


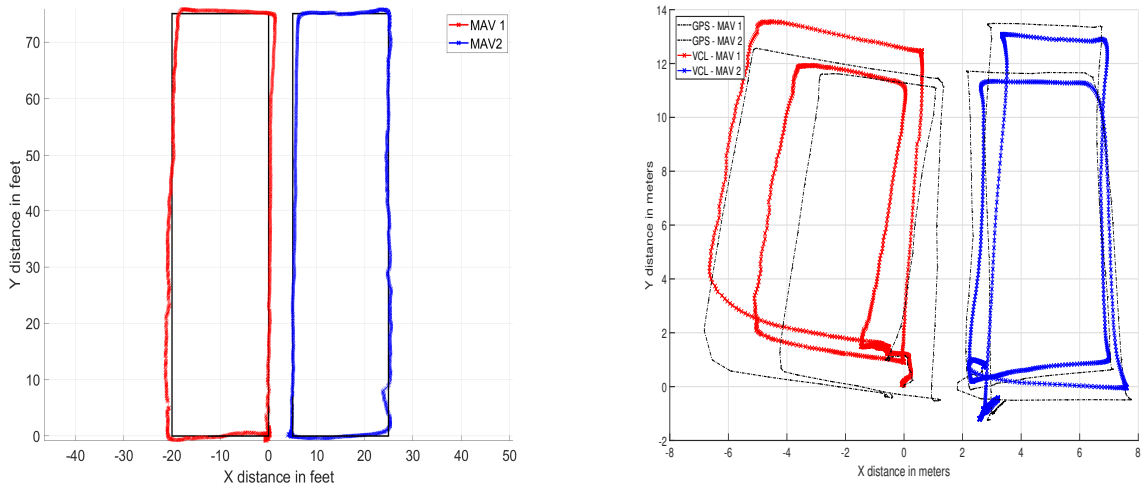
Figure 3.12: VCL position estimates for three MAVs navigating within AirSim.

MAV/MSE	X (cm)	Y (cm)	Z (cm)
1	24.11	8.57	21.26
2	13.56	36.19	83.5
3	8.32	6.56	25.58

Table 3.1: Mean squared errors for position estimates of three MAVs in AirSim

3.12.1.2 Real experiments

Similar intra-MAV localization was also tested in real scenarios with the Bebop 2 quadrotors. At first, the vehicles collaborate to isolate common features and build a map, within which each MAV attempts to localize itself. Figure 3.13 show the localization results for rectangle shaped trajectories navigated by two MAVs. The first test (3.13(a)) was meant to evaluate the accuracy of the algorithm against ground truth, so the MAVs were moved by hand along two pre-marked rectangles of dimensions 75×20 feet each. The VCL estimates



(a) X-Y positions of two Bebop 2 quadrotors moved through a trajectory of known, marked dimensions. Ground truth plotted in black.

(b) X-Y positions of two Bebop 2 quadrotors flown through rectangular trajectories. GPS position estimates plotted in dotted black.

Figure 3.13: Intra-MAV localization in real experiments

were seen to track the ground truth fairly well, while exhibiting a slight drift at the far edges (which can be attributed to changes in feature appearance when at a distance of 75 feet from the initial location where the map was built).

The second test of the intra-MAV localization involved manual flight of two MAVs in an outdoor area near a building, and comparison with the GPS position estimates. In this test, it was observed that the VCL estimates were closer to the real trajectories taken by the MAVs than the GPS, which could be attributed to low altitudes and thereby a certain degree of inaccuracy in the GPS. The results of this test can be seen in 3.13(b).

The accuracy of the orientation estimates was also evaluated in real flights. In one particular test, the two Bebop MAVs were made to fly side to side at high speeds (up to 5 m/s), and the estimates of the roll angles were compared to the IMU data from onboard the vehicle. The VCL estimates and the IMU estimates of the angles can be seen to be close to each other through figure 3.14, demonstrating an accurate estimation of angles even through fast flights.

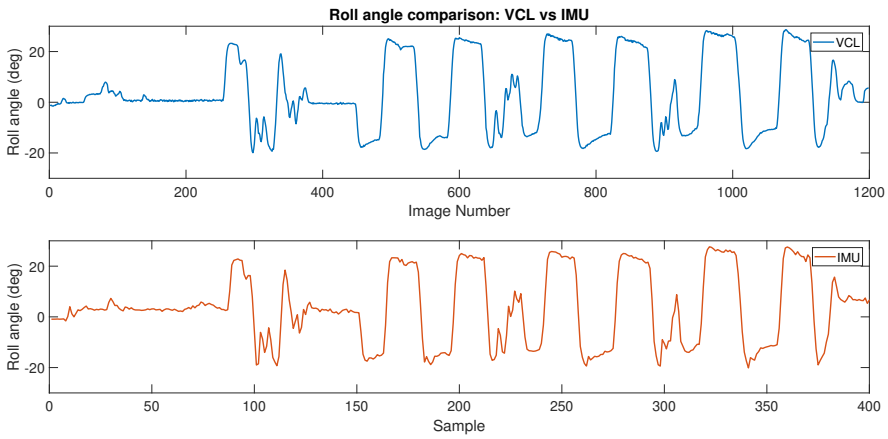
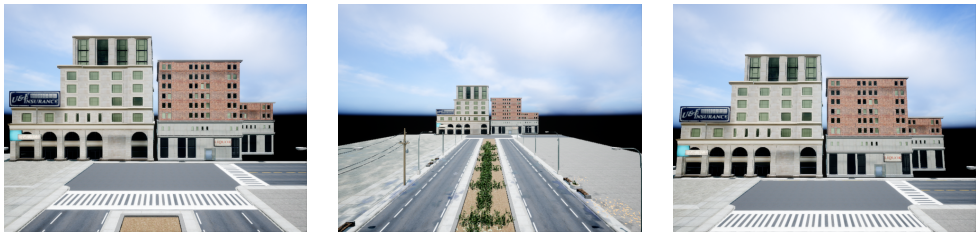


Figure 3.14: Roll angle comparison between VCL estimates and IMU for a fast side-to-side flight.



(a) Initial position **(b)** Midpoint **(c)** Final position

Figure 3.15: Inter-MAV estimation test: backward and forward trajectory sample images

3.12.2 Inter-MAV localization

3.12.2.1 Simulation

Inter-MAV localization was first tested and evaluated in the simulation, by creating a sparsely populated environment where three vehicles start off side by side near a feature-rich building, but are then commanded to fly backwards (thus away from features) and then forwards to the starting positions. Three images from this trajectory are shown in figure 3.15 to demonstrate the change in feature appearances. Between the backward and forward motion, each MAV traverses $80 + 80 = 160$ meters in the environment.

As the biggest advantage of inter-MAV localization is when one vehicle has better localization than the others, such a condition was simulated in this experiment by making

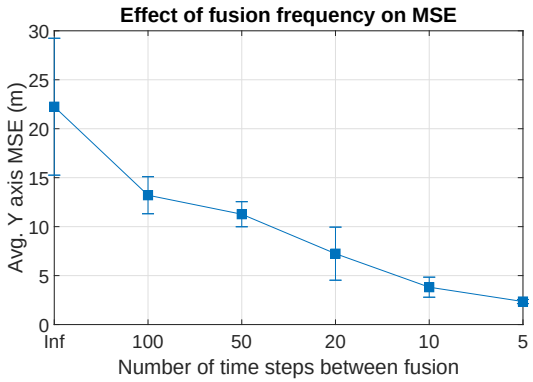
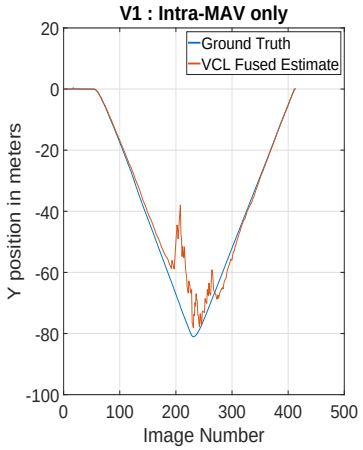


Figure 3.16: Effect of frequency of inter-MAV data fusion on position estimate error. Error bar shows variation of Y axis position MSE between clients V1 and V2.

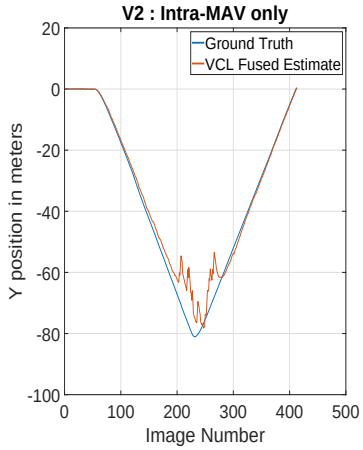
an assumption that the MAV in the center has access to better estimates. This was achieved by using the ground truth values of the positions for the MAV in the center and corrupting them with zero-mean noise in order to simulate consistently good localization. If this MAV were to be considered as the 'host', the effect of inter-MAV localization between this host MAV and the others was analyzed. Alongside, intra-MAV localization solely using the cameras was performed at every time step for both the client MAVs.

The effect of the frequency of the inter-MAV measurements can be seen visually in figures 3.17 and 3.16. This analysis focuses on the estimation error on the Y-axis where most of the movement is along. With absolutely no inter-MAV measurements, the VCL estimates show significant error at the midpoint, where the vehicles are the farthest from the scene. But once inter-MAV measurements are obtained from a MAV with higher confidence in its own position and fused, the errors decrease, and increasing the number of times inter-MAV measurements are fused decreases the error significantly. The comparison of mean squared errors for these different test cases of fusion frequency can be seen in figure 3.16. By increasing the frequency of the relative measurements to once in 5 images (approximately 1 Hz), the fused position estimates of V_1 and V_2 are much closer to the ground truth.

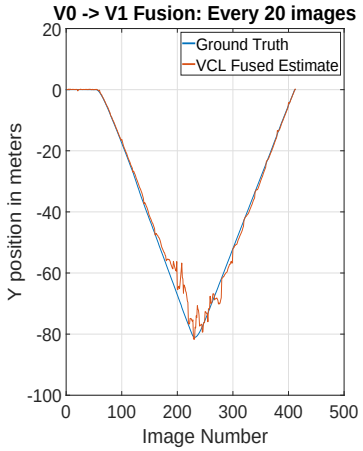
Figure 3.18 visually shows the comparison of the measurement confidences of intra-MAV measurements and the inter-MAV measurements by plotting the trace of the measurement



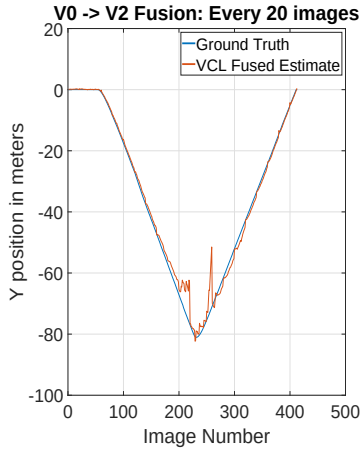
(a) Y estimate of V1 with no fusion



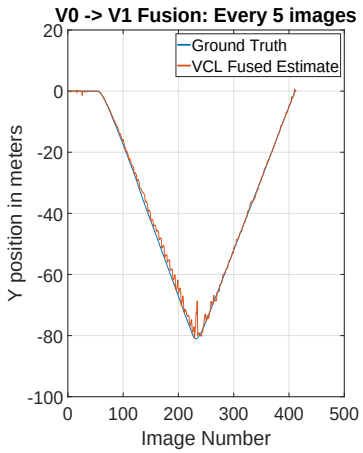
(b) Y estimate of V2 with no fusion



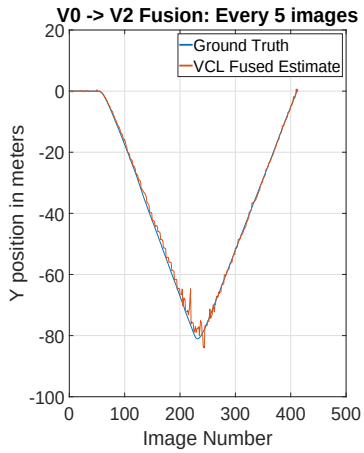
(c) Y estimate of V1 with fusion every 20 images



(d) Y estimate of V2 with fusion every 20 images



(e) Y estimate of V1 with fusion every 5 images



(f) Y estimate of V2 with fusion every 5 images

Figure 3.17: Effect of frequency of inter-MAV data fusion on mean squared error on one axis - both clients

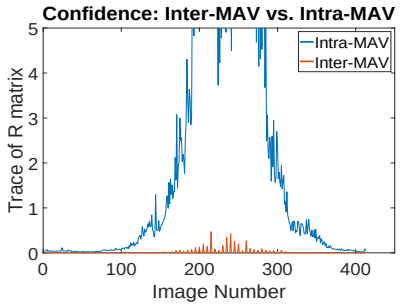
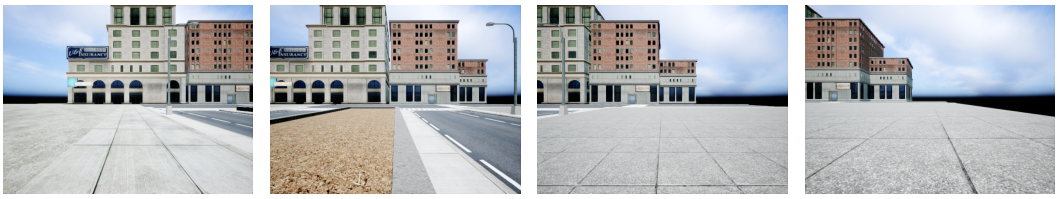


Figure 3.18: Comparison of inter vs intra-MAV measurement covariances. Inter-MAV measurements are usually seen to have significantly lower solution covariance.



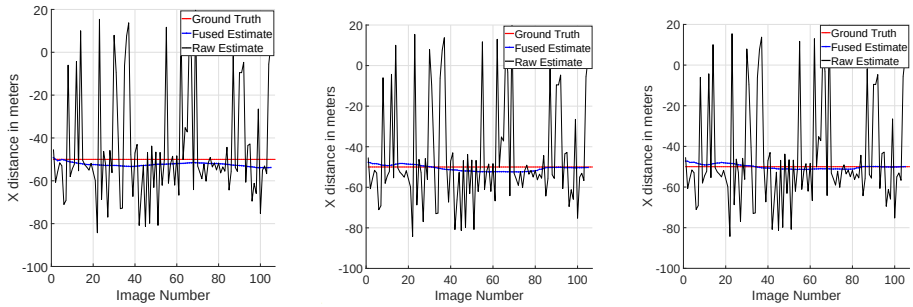
(a) V_0 **(b)** V_1 **(c)** V_2 **(d)** V_3

Figure 3.19: Images from the starting positions for four MAVs in AirSim

covariance matrix (for one MAV) over time. It is evident here that as the MAV moves away from the map features, the intra-MAV measurement covariance rises rapidly due to changes in feature appearance compared to the keyframes; but the inter-MAV covariance stays low throughout, as it only depends on the amount of feature overlap and distribution of points.

3.12.2.2 Effect of number of vehicles in group

In another experiment, the effect of the number of contributing hosts to the VCL data fusion is analyzed. For this experiment, four MAVs were placed side-by-side observing a building in simulation, spaced apart by a distance of 25m between each pair of vehicles, which is considerably higher compared to their distance from the scene. V_3 is treated as the client that needs localization assistance, because as seen from figure 3.19, V_3 has the least amount of features visible from the initial map, thus would be most prone to errors in the intra-MAV localization scheme.



(a) One host **(b)** Two hosts **(c)** Thre hosts

Figure 3.20: Client X axis position estimate with data from (a) one host (b) two hosts (c) three hosts

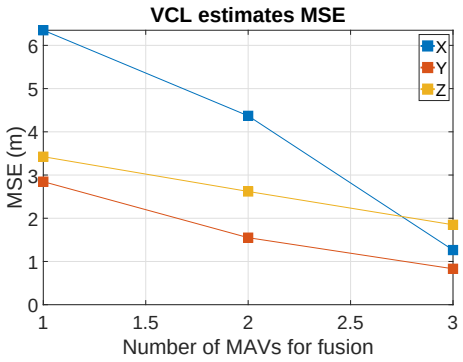


Figure 3.21: Effect of number of participants on MSE of estimates

All three hosts combining their inter-MAV estimates results in the most accurate estimate for the client V_3. In figure 3.20, the X-axis position estimates of V_3 is compared to the ground truth, with the fused estimate in blue and the raw intra-MAV estimates in dotted black. The raw estimate exhibits a very high amount of error, but the fused estimate in 3.20(c) is able to track the ground truth with an average MSE of 1.61m. The accuracy reduces with a reduction in the number of vehicles taking part in the fusion, as evidenced by the estimates in figures 3.20(a) and 3.20(b). Figure 3.21 compares the MSE on all three position axes with the number of vehicles participating in data fusion.

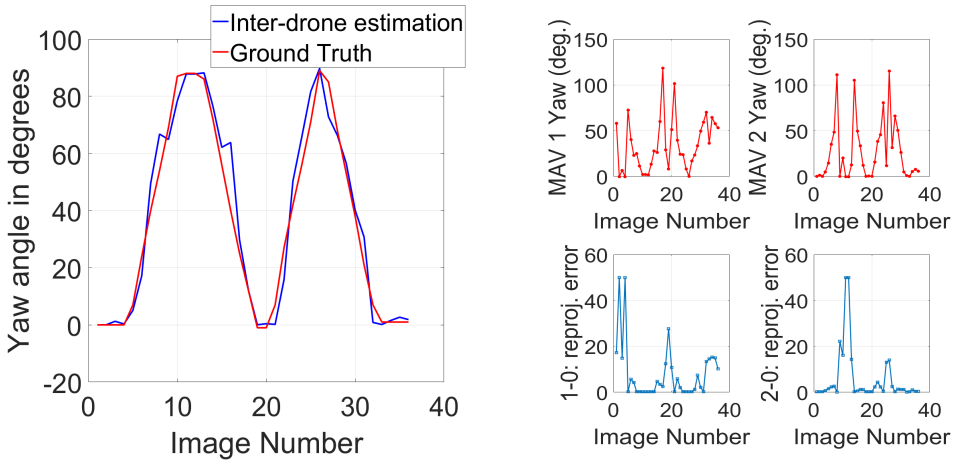
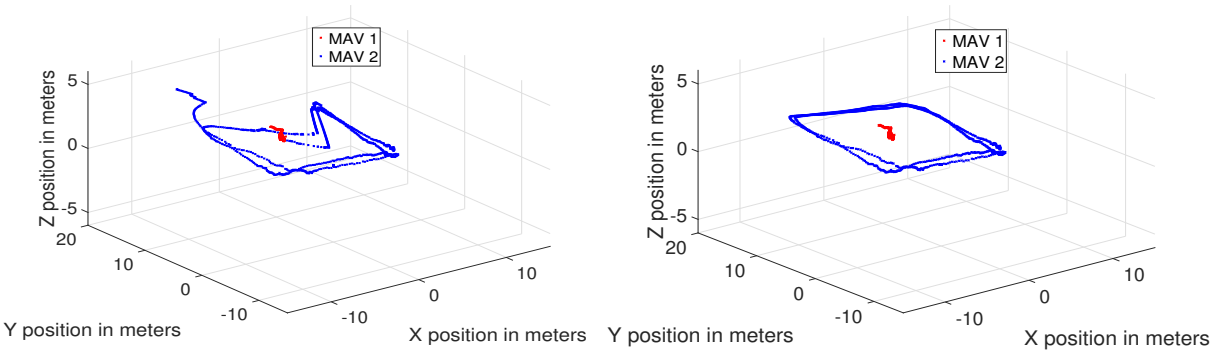


Figure 3.22: Left: Fused yaw angle estimate for a rotating MAV. Right: Yaw angle from relative estimates from the other two vehicles, and associated reprojection errors. Reprinted from Vemprala and Saripalli[64]

3.12.2.3 Handling pure rotation

One of the problems that is evident in single monocular-camera localization is the issue with pure rotation movement in the yaw direction, which is a very common maneuver for MAVs. Pure rotation usually causes existing map points to go out of view, while the fact that there is no translation by the camera means it is not possible to triangulate new feature points through a single camera without any additional information. In contrast, the VCL algorithm is able to handle this problem by capitalizing upon the inter-MAV localization feature points between what is rotating MAV and another MAV that is observing common scene points.

As a test case, an environment was simulated containing two perpendicular buildings being observed by an MAV (MAV 0) that performs periodic 90-degree rotations, trying to observe both. MAV 0's rotation speed was set to 45°/s. Two other MAVs (1 and 2) are also present in the proximity, with each of them in hover mode, observing one of the buildings from a distance. To assist with MAV 0's pose estimation during fast rotations, inter-MAV localization is exploited. For every image captured by these three MAVs, relative poses are computed between vehicles 1-0 and 2-0 and fusion is attempted with the estimate onboard



(a) Intra-MAV localization only **(b)** Inter+Intra fused localization

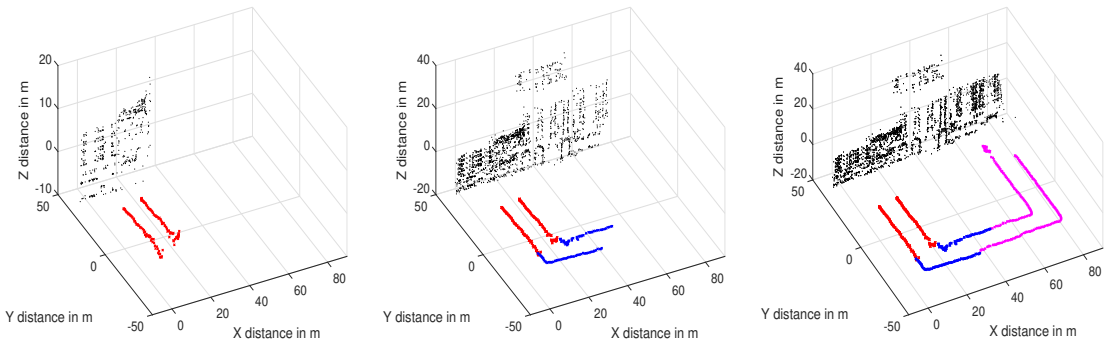
Figure 3.23: Test of fused vs unfused localization for an indoor trajectory using real MAVs

MAV 0 itself: and based on which estimate has a lower uncertainty, the final yaw angle of the rotating MAV is computed. In figure 3.22(a), the fused estimates of the yaw angle are shown, where it can be seen that the estimated angle closely matches the ground truth. A careful analysis of figure 3.22(b) shows how MAV 1 or MAV 2 is chosen as the dominant source of the yaw information based on the reprojection error (on which the uncertainty estimate depends). Swift rotations such as this are typically problematic for single-camera localization, but the collaborative aspect can maintain localization through data fusion from other, possibly more reliable, sources of information.

3.12.3 Inter-MAV localization: real experiments

After validating the efficacy of inter-MAV localization and data fusion in simulation, the VCL algorithm was also tested on data from real flights. In this experiment, two Bebop quadrotors V_1 and V_2 were flown indoors, where both vehicles first obtained initial images to map a room, and then V_1 was commanded to hover in the middle of the room, whereas V_2 navigated a square trajectory around the first. This trajectory was traversed in a way that V_2 has to view relatively less feature rich areas of the room as part of it. As a result of this, relying solely on intra-MAV localization resulted in a high amount of drift in certain areas, as can be see in figure 3.23(a).

To counter this effect, the VCL algorithm was made to perform inter-MAV localization



(a) Step 1 **(b)** Step 2 **(c)** Step 3

Figure 3.24: Process of map updates while maintaining localization: newly captured features are appended to the global map when the number of tracked features becomes low.

with V_1 acting as the host, whenever the number of features tracked by V_2 fell under a threshold (30 inliers in this case). The covariance of inter-MAV localization being much lower, it took over whenever intra-MAV localization suffered in accuracy, resulting in a significantly more accurate trajectory estimate for V_2, shown in 3.23(b).

The principles behind relative pose estimation and scale recovery can also be used for performing map updates for the whole group, when the navigation is over large spaces. Figure 3.24 shows the process of map updates as two MAVs navigate an environment in AirSim.

3.13 Algorithm requirements

Although the algorithm was not applied in real-time onboard the vehicles, some analysis was performed on the possible computational and communication requirements. Intra-MAV localization is expected to be performed on each MAV individually, whenever a new image is received, whereas inter-MAV localization is only performed in an on-demand fashion. In order to keep communication bandwidth requirements low, the algorithm was designed in a way that it does not require images to be transferred between vehicles whenever inter-MAV localization is performed. Instead, when a client vehicle needs inter-MAV localization data from a host, a packet containing the feature keypoint locations and descriptors from the

image obtained by the client is transmitted to the host. The size required per 'feature' would be a sum of the size for storing a single keypoint location: two pixel coordinates, thus a combination of two double precision numbers and thus 16 bytes and a 512-bit descriptor vector, so 64 bytes. The total requirement per feature is under 100 bytes, which stays the same for both CPU and GPU implementations of the feature algorithms. For a typical image, the total information that needs to be transferred would be in the order of kilobytes. After the host vehicles finishes its computation of the client's pose, the pose data transmission involves sending six double precision values: three positions and three orientations, again only about 50 bytes, through a simple $\mathcal{O}(1)$ update (per host). When maps are first generated, or subsequently updated, the data that has to be transmitted to all vehicles involves a set of 3D locations (24 bytes per point) and a descriptor for each location to facilitate feature tracking (64 bytes per point). This can be broadcast to all the vehicles, along with a signal indicating when it is updated, so all vehicles can read it and update their local copies.

Delays in communication were not explicitly considered in the formulation of this problem, but it is possible to construct a simple way of adapting to delays. If an inter-MAV localization process is delayed but received at a later time, it would be possible to continue localizing using intra-MAV measurements, but when the inter-MAV measurement is received, the system can back up to the previous time, update and then propagate to the present. This would involve keeping track of not only the posterior belief of the Kalman filter, but also the measurements. But because the measurements are considered to be essentially just the state vector (pose only, the map points are not part of the observations), the space requirements for storing these would be significantly lower.

4. COLLABORATIVE UNCERTAINTY-AWARE PLANNING

In the previous chapter, the problem of collaborative localization was considered for vision based collaborative MAVs, with the solution attempt examining the idea of a feature-based decentralized algorithm that combines individual pose estimation with relative pose estimation for increased accuracy. As an extension, this chapter is dedicated towards a discussion of how path planning can be performed for a group of MAVs that can localize through the VCL framework.

While path planning usually concerns the idea of connecting start and goal locations through valid paths, considering the challenges of micro aerial vehicles with regards to energy expenditure and their instability, it is important to also focus on other factors such as the uncertainty of the pose estimation or the cost of path traversal when navigating from one point to another. The former is the primary motivation to the problem of 'uncertainty-aware' path planning, which seeks to evaluate paths based on how informative they are, and thereby avoiding areas that can affect localization adversely.

As mentioned in earlier discussion, the complete planning problem is known to be computationally intractable, which is usually resolved to some extent by employing approximate approaches such as, for example, sampling. Nevertheless, for a multi-robot scenario, the dimensionality and complexity of the problem increases exponentially with the number of robots, which can still present an intractable worse case complexity when it is treated a joint problem. To offset this problem, the framework used in this work attempts to also treat uncertainty-aware planning in a slightly decentralized way by identifying two separate causes of uncertainty for vision-based approaches and addressing them in a decoupled manner.

If any vehicle is able to localize purely through vision: specifically, a feature-based

framework such as in VCL, two main factors affecting uncertainty can be identified. The fidelity of measurements obtained in a keyframe-based localization framework depend both on the quality of the features that comprise the map being localized against, as well as the quality of tracking of the features from the map as the vehicle is in motion. Hence, a planning algorithm seeking to solve the planning problem in a way that improves feature-based localization needs to improve both of these factors. Thus, the solution to the vision based collaborative path planning (VCP) problem is approached from two separate directions, which are:

1. *Map improvement*: Prior to generating and executing individual trajectories for the MAVs in a group, it is beneficial to consider the possibility of improving an existing global map. In many instances, an insufficient feature overlap, or bad initial placement of the MAVs can result in a sparse map that can adversely affect localization for all members of the MAV team. Given the multiple view geometry-inspired approach for mapping that the VCL framework uses, map improvement can be redefined as a view selection problem: to identify the most beneficial camera viewpoints for the multi-view reconstruction process which can achieve the highest possible accuracy and density of reconstruction. While generating these viewpoints, it is essential to also take into account constraints such as energy expenditure.

   In the field of computer vision, the problem of seeking an optimal placement for a single vision sensor to improve an existing representation is known as the 'next best view' (NBV) problem. NBV can be considered to be an incremental approach to build a sensing strategy for exploration or reconstruction. As an extension, active model improvement is a sub-problem that is defined as the sequential process of systematically increasing the precision and fidelity of an estimated 3D model. In the path planning framework that is the focus of this chapter, the map improvement phase can be defined as a multi-camera extension to active model improvement, where the view selection problem is solved simultaneously for multiple cameras.

2. *Uncertainty-aware planning*: If the previous step is successful, the MAVs potentially have access to an improved global map. The next step would be to plan paths from start to goal state in a localization-aware fashion as they attempt to track existing features. The VCP framework solves this step using a sampling based planner that attempts to improve vision based localization constraints: specifically those that would affect a feature-based localization scheme, thus attempting to connect only those viewpoints that optimize visibility of features along with path cost. The approach used here is inspired by previous sampling based planners for uncertainty-aware planning such as information-rich rapidly-exploring random tree (I-RRT)[50] and rapidly-exploring random belief trees (RRBT).[49]

Through the VCP framework, the MAVs are expected to collaborate in the beginning to improve an existing reconstruction, and then plan paths in a decentralized fashion. This decentralized path planning, which is expected to be performed by the MAVs individually, is responsible for constructing uncertainty-aware paths that can ensure that the MAV is in proximity to texture-rich areas wherever possible, and is continuously able to localize through feature observations.

4.1 Next best view planning for multiple vehicles

Finding the best placement of a vision sensor in order to improve an existing representation of a scene (under certain predefined constraints) is termed as the 'next best view' (NBV) problem.[89] Depending on the context, NBV can be applied for various types of vision sensors such as monocular cameras, stereo cameras and furthermore. In the VCP application, the NBV problem is applied to multiple MAVs simultaneously, (equipped with monocular cameras) and from now on, is referred to as next-best multi-view (NBMV). Given n MAVs, and a set of starting positions, it is assumed that the very first set of images acquired from the starting positions afford partial knowledge of a scene through a point cloud. Using this initial map as seed data, the NBMV problem is expressed as the problem

of calculating vehicle poses $\mathbf{x_i}$ for all $i \in [1, n]$ such that certain parameters relating to the visibility of the existing features are optimized further. This can be represented as a minimization problem for a set of vehicles over a space of candidate poses:

$$\min_{\mathbf{x_i}} f(\mathbf{x}, \mathbf{X}) \tag{4.1}$$

where f is a cost function that encodes various parameters relating to 3D reconstruction and $\mathbf{X}$ is a set of 3D points representing the scene. For the sake of simplicity, candidate poses for each MAV/sensor are selected only in 4DoF: i.e., 3D position (x, y, z) and the yaw angle ψ. Hence, a vehicle's pose forming $\mathbf{x}_i$ can be represented as:

$$\mathbf{x_i} = [\mathbf{p_i}, \psi_\mathbf{i}] | \mathbf{p_i} \in \mathbb{R}^3, \psi_i \in SO(3) \tag{4.2}$$

The operation of this NBMV planner can thus be divided into the following steps:

1. The inputs, a partial 3D scene along with the initial poses of the vehicles that resulted in this scene, are read into memory by the algorithm.

2. Several candidate viewpoint sets, each set containing n position and yaw samples for n vehicles are sampled in the valid space. Each such viewpoint set is evaluated using a custom cost function to obtain a value representing the quality of reconstruction, if one were attempted from that sample.

3. Over several iterations, the optimization routine evaluates multiple candidate samples and picks the sample that affords the least cost. Over the course of time, the algorithm is expected to converge to a locally optimal solution.

The representation of the scene considered here, in accordance to the VCL algorithm, is a 3D point cloud. A new viewpoint set that is sought out by the NBMV planner is expected to find a balance between reduction of geometric uncertainty of the reconstruction and maximization of the number of points in the point cloud. As discussed earlier, the

localization algorithm does not have access to vertex-wise 3D uncertainty estimates; hence, the quality of the map is described through a set of custom-designed criteria. These criteria are essentially a set of heuristics that are crafted from the theory of image based reconstruction and multiple-view geometry, and try to encode factors that are responsible for accuracy and quality of reconstruction for a multi-view system, and contribute systematically to the attempt of reducing uncertainty of the point cloud and subsequent localization that uses this point cloud.

4.1.1 Heuristics for optimization

To recap, the mapping phase involves image captures from each MAV at a specific location in 3D space and isolating common features between all of them. Each MAV pose x_i represented as above has a 3D-2D projection associated with it arising from the image captured from that pose. When an initial set of 3D points $\mathbf{X}$ is known, projections of $\mathbf{X}$ on an image plane can be computed trivially from any pose in 3D space. Given this information,a set of heuristics can be computed for any arbitrary pose, which when combined, describe the 'quality' of an image captured from that pose, and subsequently the quality of 3D reconstruction or localization. The heuristics are listed as follows:

1. *Visibility*: The very first constraint that needs to be satisfied is that the cameras need to keep a maximum amount of the existing features in view, while searching for more optimal viewpoint locations. The visibility factor expresses the ratio of visible features from a candidate viewpoint to the number of actual features that are part of the global map. When applied to a group of MAVs, this factor ensures that the features are visible at least from two of the members together to allow for reconstruction, while preferring a maximum number of vehicles viewing the features simultaneously. Through this last part, overlap is ensured between cameras.

2. *Span*: Vision-based algorithms such as the PNP algorithm, or 5-point algorithm have certain degenerate cases where they fail if all the points are on a straight line, or in a

small part of the image. It is beneficial to have the features spanning a large part of the image in order to obtain accurate model estimates. The span factors computes the ratio of the area currently occupied by the features in the projection from a candidate pose to the total area of the image. Preference is given to viewpoints that have a large area of the image filled with features.

3. *Baseline*: Multiple-view geometry states that when attempting to observe a scene at a specific depth Z, the baseline between the cameras b is proportional to the amount of error in depth perception. Hence, the cameras must be arranged according to their distance to the observed scene geometry (depth). This factor calculates the deviation of the baseline-to-depth ratio from a desired value. Configurations with a ratio far away from the desired value are penalized.

4. *Vergence angle*: Another factor in multi-view configurations is the vergence angle, or the angle between the projected rays from the cameras at their intersection point in the scene. When the vergence angle is high, maximum overlap is guaranteed but at the expense of matching error because of the different appearance of the features in both the views. A lower vergence angle is preferred wherever possible, while still being in agreement with factors such as baseline and overlap.

5. *Collisions and occlusion*: Configurations that can potentially result in collisions between the vehicles or the scene (distance between vehicles below a certain threshold) and occlusion (one vehicle blocking the view of another) are penalized.

The cost function that needs to be optimized is expressed as a weighted combination of the above metrics, where a minimum value indicates the best set of viewpoints. This combined optimization function takes a set of candidate poses for all vehicles being used as its input and outputs a value of the objective function that encodes all the heuristics.

Apart from the vision heuristics listed above, it is important that the cost function in the NBMV planner also includes the path cost for every evaluated sample: which is the sum

of the total Euclidean distance traveled by the MAVs to reach that particular sample. This path cost is combined with a weighting parameter δ, which determines how much priority is assigned to minimizing the path cost. High values of δ constrain the amount of distance the MAVs are allowed to travel in search of optimal viewpoints , in comparison with the reduction of the cost function.

For a given sampling space, the behavior of this cost function can be complex. As evidenced by the fact that the function's value depends on the number of MAVs in the team, the function can live in a high-dimensional space, while also depending heavily on scene geometry and being sensitive to small changes in pose parameters such as the yaw angle. This makes it hard to estimate an algebraic model of the function and to evaluate the derivatives in order to understand the direction and location of the minimum: which is typically required for common approaches such as gradient descent. Given this constraint, the NBMV planner considers the minimization of this cost function as a black-box optimization problem and uses a derivative-free method known as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to solve it.

4.1.2 CMA-ES optimization

The CMA-ES algorithm, first presented in,[90] with numerous updates and improvements afterward, is an evolutionary algorithm that is suited for high dimensional black box optimization problems. CMA-ES typically works on bounded or unbounded constraint optimization problems and is well suited for functions that exhibit highly nonlinear behavior. Especially for cases where derivative methods such as quasi-Newton BFGS or conjugate gradient may fail due to discontinuities, sharp ridges etc., CMA-ES employs an alternative approach by not considering or using approximate gradients. Thus, this method is feasible for even non-smooth or non-continuous functions.

CMA-ES is an evolutionary algorithm with a second order approach. Similar to other evolutionary algorithms such as the genetic algorithm, it works on the concept of a 'population', which is first initialized as a set of random samples, and subsequently modified

through a mutation step, resulting in new generations. This combination of samples can be treated as a random distribution, and this distribution is 'adapted' between iterations through a sampling step. The basic equation for sampling looks as below at generation $g+1$:

$$\mathbf{x_k}^{g+1} = \mathbf{m}^g + \sigma^g \mathcal{N}(0, \mathbf{\Sigma}^g) \tag{4.3}$$

Here, σ denotes a step size: indicating the movement of the samples between iterations, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ represents the covariance matrix of the distribution, which essentially characterizes the shape of the distribution ellipsoid. The primary strategy for driving the evolution is by estimating and evaluating the covariance matrix of the distribution. The algorithm determines the strongest sample by evaluating the function at all these samples; and once these values are available, the mean of the distribution is updated such that the likelihood of strong samples is maximized in the next step. Likewise, the covariance matrix of this distribution is also updated at every iteration, such that the direction of the sampling is biased towards stronger solutions. CMA-ES uses various ways of updating the covariance matrix, all adhering to the idea that adaptation should increase the likelihood of successful steps. In a covariance matrix, the off-diagonal terms represent the dependences between the variables while evaluating the distribution: in a case like next-best-view where the variables are tightly coupled in their effect on the final cost function, CMA-ES learns the pairwise dependencies between variables effectively through its adaptation strategy, and eventually, how changes in these variables affect the cost function.

4.1.3 Working of the NBMV planner

The NBMV planner involves initializing the CMA-ES algorithm with the initial poses (position and yaw) of the MAVs as the seed variables. A population of samples is gathered around these positions and the algorithm iteratively searches for the best viewpoints, returning the solution after a fixed number of iterations. This solution is then broken down into position/yaw for individual vehicles and supplied to them as waypoints. The work in

this dissertation focuses mainly on evaluation of these solutions and examining the effect of the NBMV algorithm's solutions on the reconstruction/localization quality, and the problem of how to actually navigate to the waypoints is not considered directly. For a system that uses collaborative localization in real time, these waypoints would have to be coupled with a feedback control system to ensure they actually reach the specified waypoints. Given any position errors and drift, it is entirely possible that the MAVs fail to reach the exact commanded viewpoint locations. But such localization drift can be offset by the fact that the positions reported by the MAVs can be re-evaluated within the NBMV algorithm, and if they do not match the expected minimum value of the cost, they can be controlled in the right direction in an iterative fashion by the controller until the cost function is close enough to the expected minimum value.

Algorithm 2 Next Best Multi View Algorithm

procedure COMPUTENBMV(x, X, C)
    set λ ▷ Population size
    initialize $m \leftarrow x$, σ, $C \leftarrow I$
    while $\neg stop$ **do**
        for $i \in [1, \lambda]$ **do**
            $x_i \leftarrow \mathcal{N}(m, \sigma^2 C)$
            $f_i \leftarrow$ computeHeuristics(x_i, X)
        end for
        $x_{1...\lambda} \leftarrow x_{f(1)...f(\lambda)}$
        $C \leftarrow$ updateCovariance()
        $\sigma \leftarrow$ updateStepSize()
    end while
    return x
end procedure

4.2 Localization aware path planning

As the MAVs in this context rely solely upon feature based localization, a planner attempting to be localization-aware while planning paths to move from any start to goal

locations needs to optimize certain factors relating to monocular vision and reduce the uncertainty arising from the localization. It must be recalled here that the intra-MAV localization, the fundamental way of localization, uses the PNP algorithm and relies on 3D-2D correspondences to estimate the pose of the MAV. Hence, the uncertainty in localization depends mainly on the visibility and appearance of the map features on the image, along with other relatively minor factors such as the angle of viewing, distance from the area where the keyframes were captured from etc. Optimizing these features can contribute to a reduction in the measurement uncertainty, and thereby the pose covariance of the vehicle itself. In this discussion, only one vehicle is considered to be performing uncertainty-aware path planning.

Assume the space that the MAV is traversing as a bounded set. Given such a bounded open set $X \subset \mathbb{R}^d$ partitioned into an obstacle region X_{obs} and an obstacle-free region X_{free}, the path planning problem is to find a path $\pi : [0, T] \in X_{free}$ that is collision free at all points and satisfies both the initial and goal state constraints. While many such feasible paths may exist, the concern of a vision-aware algorithm is to find a path that ensures vision based localization, to reduce uncertainty along the path, while also attempting to minimize path cost as much as possible. The path cost is expressed as a Euclidean distance metric from the start to any current position in space.

The space X is also supposed to contain a known number of 3D points, which comprise the global map, a set represented by M. The vehicle navigating in space X is assumed to be equipped with a camera C which is characterized by a projection matrix $\mathbf{P}$. As discussed earlier, this camera is capable of estimating 2D projections of any subset of M on the image plane. These projected features are the ones that are utilized by the PNP algorithm in order to localize the vehicle in space. Any possible location for the vehicle in X_{free}, can thus be characterized by a metric of 'information': representing the quality of the measurements of the camera projections of the 3D map points. Now, the objective can be expressed as a combination of minimizing the cost of the traversed path while maximizing the information

gain.

For the general path planning problem, sampling-based methods, which seek approximate connectivity in X_{free} by randomly sampling configurations and checking feasibility, have been found to have several desirable properties. Sampling based planners are suitable for application in high-dimensional spaces, while affording performance that scales well with available computational resources. The RRT (rapidly-exploring random tree), has particularly been a baseline for several planning approaches because of its ability to generalize over various types of vehicle dynamics. For the VCP framework, a customized version of RRT* named Vision-Aware RRT* (VA-RRT*) was implemented, which is described in detail below.

4.2.1 Vision-Aware RRT*

The modus operandi of a general RRT is to construct and maintain a tree-structured graph $\mathcal{T}$, where various nodes sampled in the state space of the MAV act as the vertices of the graph and the connecting segments between these nodes form the edges of the graph. Similar to the discussion regarding the NBMV planner, in this discussion, the state space is limited to four dimensions: position and yaw. Thus, each sample, or 'node': consists of these four values. The tree originates from a start node q_{start} (also the initial state of the vehicle) and attempts to connect X_{free} through various edges, which are in turn connections between these random samples. The lengths of segments that form the edges can be limited by a quantity known as the step size, which controls the maximum length of a connection between two nodes. In the VA-RRT*, each node $q \in \mathcal{T}$ represents a tuple $q_i = \{x_i, c_i, \sigma_i, \mathcal{I}_i, \mathbf{\Sigma_i}, J_i\}$. The various components of this node description will be discussed later.

4.2.1.1 Tree expansion

In every iteration of the algorithm, the first step of VA-RRT* is to generate a valid random sample q_{rand} from X_{free}. Any sample can be checked for possible collisions with

known objects in the 3D space, thus ensuring it is part of X_{free}. Expansion of the tree typically involves finding the nearest neighbor q_{near} to q_{rand}, and this is performed using a Euclidean distance metric. Next, a steering function is implemented that seeks to connect q_{near} to q_{rand} but while adhering to the preset step size that the tree can move in a given iteration. The result of this operation is a new node q_{new}. If the distance between q_{near} and q_{rand} is already less than the specified step size, q_{new} would be the same as q_{near}.

For a vehicle using feature-based localization, not every sample in X_{free} guarantees localization ability. There can exist numerous configurations that are collision-free, but could be bad for the vehicle either due to a complete lack of visibility of the map feature, or a bad distribution of features in the image, which can render the vehicle unable to localize and cause drift. Hence, VA-RRT* uses a vision-aware sampling strategy. Once a new node is sampled and steering is attempted, the camera projections of the map points are computed from q_{new}. If this results in a zero visibility of map points, this node is discarded and the algorithm attempts the sampling step again. Else, this new node is added to the tree and the algorithm proceeds with node evaluation. The subset of samples that guarantee a non-zero visibility of map points can be considered to be $X_{loc} \subset X_{free}$.

4.2.1.2 Uncertainty metrics

For an MAV navigating in $X_{loc} \subset X_{free}$, observations z_i are generated according to a specific observation model for every newly sampled and connected state. Given a potential path π, it would be ideal to assess the localization quality of this path via the posterior belief of the vehicle, which would be a function of the set of observations received throughout the path $\mathbf{Z}_\pi$. But, as planning is a high level process that is typically executed prior to navigation, it is impossible to anticipate the true measurements and the exact sequence of observations affecting the belief of the vehicle at this phase. Hence, the objective of an information-based planning approach is to approximate these potential observations and their quality and embed the expected visual information quantitatively into the existing graph: thus simulating the potential uncertainty expected at nodes within the planning

problem.

According to this idea, while constructing the tree, VA-RRT* also attempts to describe each node through an uncertainty metric, which represents the quality of that node in the sense of vision based localization. First, every node that needs to be evaluated is assigned a set of heuristics, similar to the NBMV problem but more specific to single-camera localization. The heuristics are listed below, and are calculated every time camera projections are computed for a valid node.

1. Visibility: The first heuristic computes the ratio of the number of points in the projected image that can be tracked given a specific state and the total number of points in the global map.

2. Span: The second heuristic ensures that images captured from locations that are close to the existing map features are assigned higher confidence. The calculation of this heuristic involves a binning operation on the image (the number of bins can be selected prior) and counting how many bins contain points from the map.

3. Viewing angle: Feature-based localization algorithms such as the perspective-N-point as well as several tracking algorithms exhibit better performance when the features are viewed at a low angle of vergence. Ideally, fronto-parallel views of a known scene result in the best localization, and to simulate this effect, the viewing angle is described through the cosine to include it in the heuristic metric.

A weighted combination of these metrics is encoded in a value $q.\sigma$ for each node, and is representative of the quality of measurements that can potentially be obtained from that particular sample.

4.2.1.3 Uncertainty propagation

Once a metric is available to express a simulated uncertainty from a given sample, it is necessary to approximate the accumulation of good measurements through edges using a

measure of information. This is because while measurement uncertainty is discrete and has a unique value for each sample, the actual pose uncertainty at a given sample is always a function of the path taken to reach that sample. Hence, each pose in the sample space can theoretically have multiple uncertainty values, each corresponding to a unique path taken to reach the sample. Some of the common ways of quantifying information are mutual information, divergence and Fisher information. Of these methods, typically mutual information and divergence require approximating the posterior belief. In the case of the VCP framework, no complex system dynamics are considered for simplicity, hence it is possible to opt for a simpler metric. Thus, the Fisher information based metric was chosen to develop uncertainty propagation inspired by the method presented in the information-rich RRT.[50] Fisher information is also a measurement-free informativeness metric, which can work solely with the information of map points and simulated 2D projections from the samples.

The Fisher information matrix $\mathcal{I}$ describes the 'information' contained by a sequence of measurements $\mathbf{z}$ about the estimation of a quantity $\mathbf{x}$. The advantage of using the FIM as a metric for uncertainty is that the inverse of a true FIM directly represents the achievable lower bound on the estimate covariance, which is the Cramer-Rao lower bound (CRLB). It can be recalled from the Kalman filter related discussion in section 3.9 that a discrete system with linear state transitions and measurements can be modelled within a recursive estimation framework as:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \tag{4.4}$$

$$\mathbf{z_k} = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{4.5}$$

The zero-mean Gaussian white noises $\mathbf{w}$ and $\mathbf{v}$ are described by covariance matrices $\mathbf{Q_k}$ and $\mathbf{R_k}$ respectively. When attempting to propagate the uncertainty from one instant to another (which can be encoded by one edge in the tree), the conventional recursive

estimation equation provides this relationship between the start and end state covariance matrices:

$$\begin{aligned}\mathbf{P}_k &= (\mathbf{I} - \mathbf{K_k}\mathbf{H_k})\mathbf{P_{k-1}} \\ &= (\mathbf{I} - (\mathbf{P_{k-1}} * \mathbf{H_k^\top}(\mathbf{H_k}\mathbf{P_{k-1}}\mathbf{H_k^\top} + \mathbf{R_k})^{-\mathbf{1}})\mathbf{H_k}) * \mathbf{P_{k-1}} \\ &= \mathbf{P_{k-1}} - \mathbf{P_{k-1}}\mathbf{H_k^\top}(\mathbf{H_k}\mathbf{P_{k-1}}\mathbf{H_k^\top} + \mathbf{R_k})^{-\mathbf{1}}\mathbf{H_k}\mathbf{P_{k-1}}\end{aligned} \tag{4.6}$$

The expression in equation 4.6 can be subject to the Woodbury matrix identity, which allows an alternative representation of the covariance computation as

$$\mathbf{P_k^{-1}} = \mathbf{P_{k-1}^{-1}} + \mathbf{H_k^\top}\mathbf{R_k^{-1}}\mathbf{H_k} \tag{4.7}$$

This inverse of a covariance matrix is equivalent to the Fisher information matrix. At the same time, it has to be noted that the measurements being acquired, either by the PNP algorithm in the VCL framework or these simulated measurements within VA-RRT* are measurements of the state directly, which means the measurement Jacobian $\mathbf{H}$ is an identity matrix. Hence, equation 4.7 can be rewritten as follows.

$$\mathcal{I}_k = \mathcal{I}_{k-1} + \mathbf{R}_k^{-1} \tag{4.8}$$

Equation 4.8 points out a critical property of the Fisher information matrix in the realm of uncertainty propagation. The FIM exhibits a simple additive property,and the information content on a set of nodes (a path) would be just the sum of all individual FIMs in the set, thus the sum of all inverse measurement noise covariances. This property enables a computationally efficient approach of propagating uncertainty within the VA-RRT*. For any node k, the FIM at that node is formed through a recursive update of FIMs through its parent $k-1$. Naturally, maximizing the information matrix is equivalent to minimizing the state covariance. Hence, in order to represent the uncertainty metric that needs to be minimized

over any path, VA-RRT* uses another quantity $\mathbf{\Sigma}$, which is a simulated covariance, to denote the uncertainty at any given node, as a function of the recursive information update received from its parents. This matrix $\mathbf{\Sigma}$ is not expected to be an accurate pose covariance estimate: but merely a quantity indicative of whether or not a certain node or path is conducive to good localization.

$$\mathbf{\Sigma_k} = \mathcal{I}_k^{-1} \tag{4.9}$$

4.2.1.4 Cost metric and rewiring

The VA-RRT* combines the classic RRT* effect of attempting to minimize path cost with this new representation of vision based localization ability. The localization ability is expected to be described the combination of heuristics described in the previous section: so in turn, each node that results in a successful connection within X_{free} can be described by a unique combination of path cost and estimated covariance. As it can be seen from equations 4.8 and 4.9, each connection results in a propagation of uncertainty from the parent to the child node. When a new node is sampled and described, the main function of the VA-RRT*, similar to other optimal tree-based planners is to examine whether moving from this node to any previously-connected neighbor would be more beneficial. In other words, if propagating from the current node q_{new} to another node q_{near} results in a cost value better than q_{near}'s existing cost, the edge that is already connecting q_{near} with its parent is pruned, and a new connection is made between q_{new} and q_{near}. This comparison is performed through a weighted cost function that is defined below.

$$J_i = w_c * q_i.c + w_\Sigma * q_i.\mathbf{\Sigma} \tag{4.10}$$

The path cost for the node in question represented as $q_i.c$, which is the combination of Euclidean distances over all edges that connect the start state to the node q_i. To describe the effect of covariance, the A-optimality criterion, i.e., the trace of the covariance matrix (inverse of the accumulated information matrix) is used. The parameters w_c and w_Σ

are considered to be tuning parameters. The objective function in equation 4.10 is also responsible for a cost vs. uncertainty reduction trade-off. Setting w_Σ to zero would cause the algorithm to perform similar to a normal RRT*, choosing paths based on the sole criterion of travel distance. Increasing the value of w_Σ would make the algorithm prefer covariance minimization over path cost.

To enable quick propagation of good measurements through the tree, VA-RRT* uses a concept known as cascaded rewiring, similar to the RRBT algorithm. If any node results in rewiring changes to the neighbors, the neighbors are rewired promptly and the VA-RRT* adds these newly changed nodes into a queue. The same rewiring check is repeated for newly added nodes in the queue. This queue ensures that the discovery of areas affording good measurements and higher reduction in uncertainty is updated recursively through the entire existing graph, rewiring the whole graph.

4.3 Trajectory generation

The VA-RRT* algorithm is responsible for returning a valid, collision-free path that, depending on the cost constraints, attempts to drive an MAV to texture-rich areas in the map with the assumption that its actual localization would benefit from it. However, it does not explicitly take the dynamic model of the vehicle into account and thus, the paths returned by the algorithm are essentially coarse connections through the free space of the environment. For a system such as an aerial vehicle with highly nonlinear, high degree-of-freedom dynamics, a coarse path like this can be infeasible and constructing a dynamically feasible trajectory usually involves iterating over the equations of motion.

In order to solve this problem, the VA-RRT* algorithm is combined with a minimum-snap trajectory generation scheme that was presented by Richter et al[91] as a modification of earlier UAV trajectory generation work by Mellinger et al.[62] This trajectory generation algorithm works on a combination of a 3D occupancy map and a high level path returned by the VA-RRT* to construct a sequence of polynomial segments that are jointly optimized in order to connect the VA-RRT* path nodes into a smooth trajectory between start and goal.

Algorithm 3 Vision-Aware RRT*

procedure VARRT(q_{start}, q_{goal}, X) ▷ X = map points
  init $\mathcal{T}$
  init w_c, w_Σ
  $\sigma_{init} \leftarrow$ computeHeuristics(q_{start}, X)
  $q_{start}.c = 0, q_{start}.\sigma = \sigma_{init}, q_{start}.p = -1, q_{start}.\mathbf{\Sigma} = \mathbf{\Sigma}_{init}$
  while $N_{iter} < Iter_{MAX}$ **do**
    $q_{rand} \leftarrow$ sample()
    $q_{nearest} \leftarrow$ knn($\mathcal{T}, q_{rand}$)
    $q_{new} \leftarrow$ steer($q_{nearest}, q_{rand}, \epsilon$)
    $q_{new}.\sigma, q_{new}.R \leftarrow$ computeHeuristics(q_{new}, X)
    if $q_{new}.\sigma.v == 0$ **then**
      goto 5
    end if
    $e_{new} \leftarrow$ connect(q_{near}, q_{new})
    if $q_{new}, e_{new} \notin X_{free}$ **then**
      goto 5
    end if
    $q_{new}.\mathcal{I} = q_{nearest}.\mathcal{I} + (q_{new}.R)^{-1}$
    $q_{new}.\mathbf{\Sigma} = q_{new}.\mathcal{I}^{-1}$
    $q_{new}.J = w_c * q_{new}.c + w_\Sigma * q_{new}.\mathbf{\Sigma}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup q_{new}, e_{new}$
    $Q \leftarrow Q \cup v_{new}$
    while $Q \neq \emptyset$ **do**
      $u \leftarrow pop(Q)$
      for $u_n \in$ knn(u) **do**
        $J' =$propagate($u.J, u_n$)
        if $J' < u_n.J$ **then**
          $G.E \leftarrow G.E \setminus \{u_n.p, u_n\}$
          $u_n.p = u$
          $G.E \leftarrow G.E \cup$ connect(u, u_n)
          $Q \leftarrow Q \cup u_n$
        end if
      end for
    end while
  end while
  $q_{neighbor} \leftarrow$ knn(q_{goal})
  while $q_{neighbor}.p \neq -1$ **do**
    $Path \leftarrow Path \cup q_{neighbor}.p$
    $q_{neighbor} = q_{neighbor}.p$
  end while
  return $Path$
end procedure

Minimum-snap splines have been shown to be excellent choices for multirotor vehicles, since the motor commands of these vehicles are proportional to the snap (the second derivative of acceleration) of the path. Hence, minimizing the snap of a path is equivalent to reducing abrupt movements and also helps in avoiding loss of observations.

The method for minimum snap trajectory generation involves concatenating a set of individual optimization problems, each working on one segment of the full path. Each segment is composed of trajectories for the variables of positions x, y, z and yaw ψ. A multirotor is considered to be a 'differentially flat' system: which means that the system has an output that can be used to explicitly express all states and inputs without requiring integration, so only in terms of the outputs and some of its derivatives. The differentiability of these polynomial segments makes them applicable for a differentially flat representation of quadrotor dynamics. In order to minimize the snap of these polynomial trajectories, a cost function for optimization is described as follows.

$$J = \int_0^T c_0 P(t)^2 + c_1 P'(t"^2) + c_2 P''(t)^2 + ... \tag{4.11}$$

To solve for a minimum snap trajectory, the coefficient corresponding to the fourth derivative would be the only non-zero component. This can be rewritten in matrix form as

$$J = \tilde{\mathbf{p}}^\top \mathbf{Q} \tilde{\mathbf{p}} \tag{4.12}$$

The method in Richter et al[91] which is used as part of the VCP framework, expresses this minimization problem as an unconstrained quadratic programming problem which results in an efficient computation of the final smooth trajectory.

4.4 Implementation and Results

The planning framework was validated on a combination of manually created sample scenarios and environments from the AirSim simulation platform. Open source implementations of the CMA-ES algorithm were used in both C++ and MATLAB, and the VA-RRT* was

similarly programmed in both C++ and MATLAB. The C++ implementation of the VA-RRT* uses a KD-tree for storage as well as efficient k-nearest neighbor queries during the course of the algorithm execution. The C++ versions of the NBMV and VA-RRT* algorithms were also integrated with ROS.[92] Point clouds of the map were converted to OctoMap occupancy grids,[93] and the flexible collision library (FCL)[94] was integrated within VA-RRT* in order to sample states and perform connections in a collision-free fashion. The VA-RRT* computed coarse paths were passed to the minimum-snap trajectory generation module made for ROS in order to compute a smoothed, minimum-snap trajectory feasible for navigation. During the evaluation of both NBMV and VA-RRT* modules, the focus is mainly on sparsely populated environments, as they present the biggest challenge to feature-based localization methods.

4.4.1 Next Best Multi-View Planning

A depiction of the basic working of the multi-vehicle NBMV planner is shown in figure 4.1. In this experiment, two MAVs were initialized in a sparsely populated map in AirSim (containing one building far away from the MAV positions). The initial images captured by the MAVs can be seen in 4.1.a and 4.1.b. This pair of images resulted in a sparse reconstruction (a point cloud with 83 points) which was then used as initial knowledge for the NBMV algorithm. Through sampling the valid space and examining the simulated projections, the NBMV planner was able to compute a new pair of positions, under the assumption that attempting a reconstruction from these points could result in a better map. When reconstructed from these new viewpoints, the map was updated with a point count of 207 (4.1.f), demonstrating significant improvement in the density.

To evaluate whether the generation of a new map actually improved the localization, one of the MAVs was made to execute a rectangular trajectory in this environment, which was first localized against the initial, sparse map and next against the updated, relatively denser map. The comparison of the estimation results from the VCL algorithm can be seen in figure 4.2. Localization using the map generated from the closer viewpoint resulted in estimates

(a) Initial view L **(b)** Initial view R **(c)** Optimized view L **(d)** Optimized view R **(e)** Initial point cloud **(f)** Optimized point cloud

Figure 4.1: Result of NBMV planner for a case with 2 MAVs. Initial and final maps seen in (e) and (f). Reprinted from Vemprala and Saripalli[66]

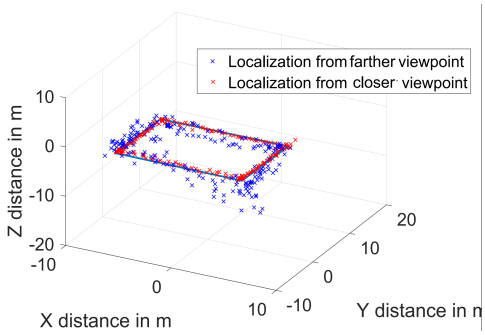


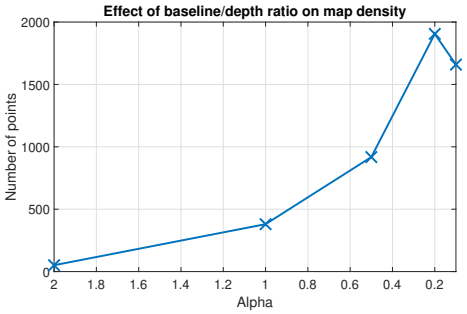
Figure 4.2: Performance of localization against an initial map and a map improved through the NBMV algorithm. Reprinted from Vemprala and Saripalli[66]

that matched the ground truth better with a mean squared error (MSE) of approximately 23 cm, whereas using the map from the farther viewpoint resulted in performance that exhibited a significant amount of drift and inaccuracy.

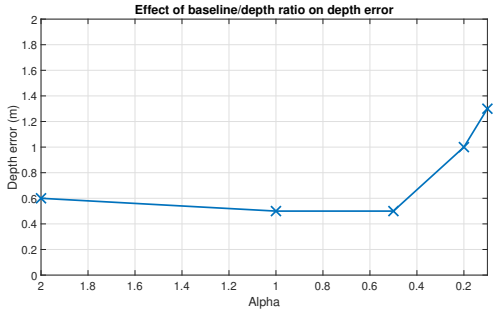
Table 4.1: Comparison of localization accuracy with maps from different runs of the NBMV planner. Reprinted from Vemprala and Saripalli[66]

Index	1	2	3	4	5	6	7	8	9	10
MSE (cm)	17.2	18.8	11.4	14.2	22.2	20.9	19.1	24.3	21.2	19.4
# pts	217	248	271	200	199	230	232	197	212	257

As CMA-ES is an evolutionary strategy, it contains a certain degree of randomness which can change the result slightly between runs, especially when multiple sets of viewpoints can result in a minimum value of the cost function. In order to test these characteristics, a repeatability test was performed, where the previous test case was repeated ten times.



(a) Baseline-depth ratio vs. point cloud density



(b) Baseline-depth ratio vs. map depth error

Figure 4.3: Effect of baseline-depth ratio on mapping quality

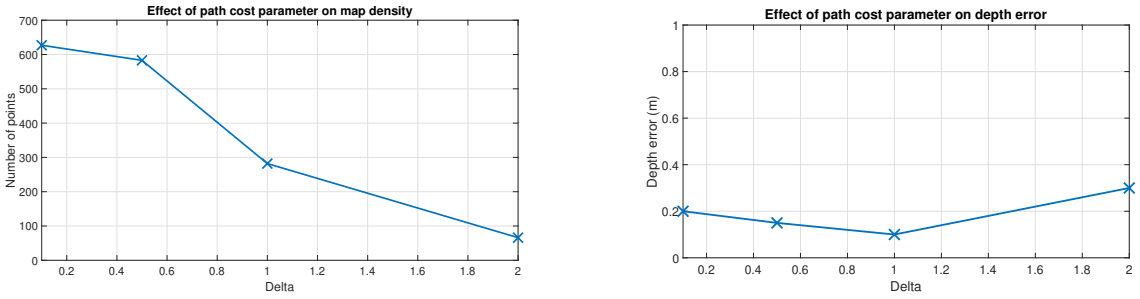
The NBMV planner was asked to compute optimal viewpoints for both the vehicles, and for each set of viewpoints, a map was reconstructed from those locations and the rectangular trajectory was localized against this map. Table 4.1 shows a comparison of the runs in terms of localization accuracy. The average position MSE over all axes compared to the ground truth was under 25 cm for all the maps, showing that the algorithm is able to consistently result in good viewpoints for map improvement.

4.4.1.1 Effect of baseline-depth ratio

Figure 4.3 depicts the analysis of the effect of the choice of α, the baseline-depth ratio on the performance of the NBMV algorithm. Constraining α to low values results in a high density of the point cloud due to maximal overlap and better feature matching, but at the expense of higher depth error. The values in figure 4.3 are purely for demonstrative purposes, as the actual characteristics of the effect of α are very much dependent on camera parameters. For general operation of the NBMV planner for the AirSim cameras, an α value of 0.5 was chosen.

4.4.1.2 Effect of path cost constraint

In figure 4.4, the effect of the choice of δ, the path cost parameter is shown. Increasing the value of δ means larger distances traveled by the MAVs in search of optimal viewpoints are penalized greatly: this results in the final locations being constrained to farther locations



(a) Path cost weight vs. point cloud density **(b)** Path cost weight vs. map depth error

Figure 4.4: Effect of path cost weight on mapping quality

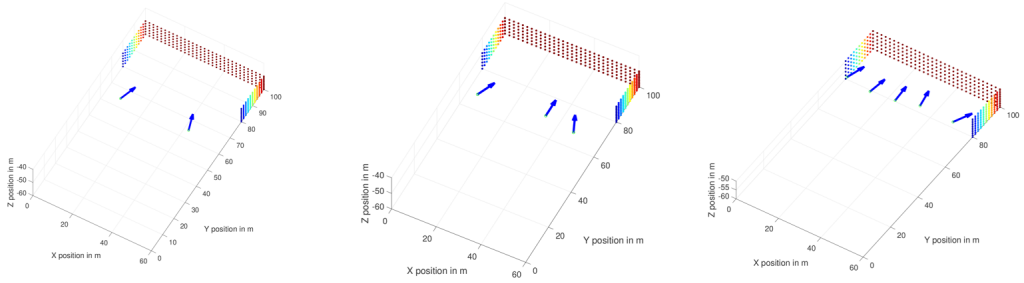


Figure 4.5: Performance of the NBMV planner when its has access to different numbers of vehicles.

from the map. But even with this constraint, the NBMV algorithm ensures that the baseline between cameras is varied accordingly in order to reduce reconstruction error. As a side effect of increasing δ, the point cloud density decreases.

4.4.1.3 Effect of number of vehicles

Extending the NBMV planner to more than two vehicles brings many advantages into the scene. The most obvious advantage is that increasing the number of cameras that are responsible for observation of the scene from many cameras can bring additional features into view, thus increasing the density of the reconstructed point cloud. Using more cameras for reconstruction also means that the baseline constraint can be relaxed, allowing the cameras to move closer to the scene being observed. Figure 4.5 shows an example where different 2, 3 and 5 cameras were run in the NBMV planner. As the number of cameras

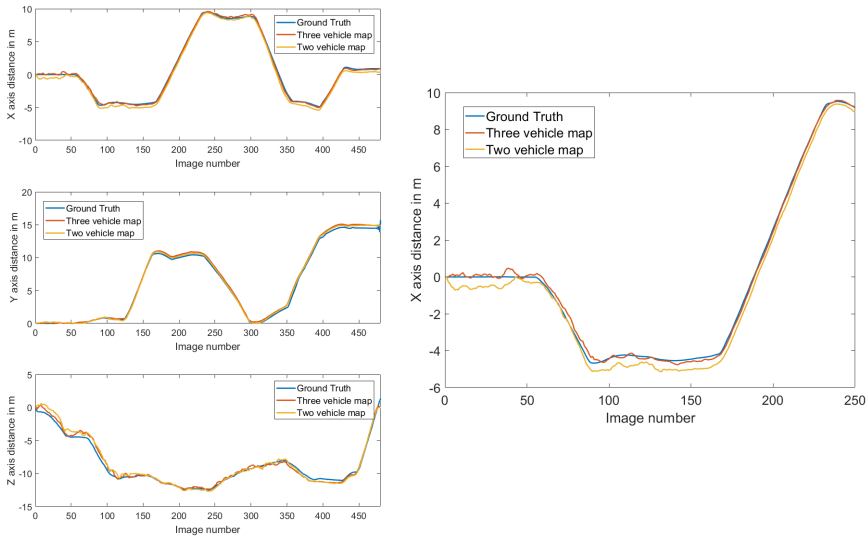


Figure 4.6: Localization error between maps made by 2 vs 3 vehicles.

increases, it can be noticed how the observation distance also decreases. A similar case of using multiple vehicles for mapping was analyzed through AirSim: and table 4.2 shows that the more cameras are used for mapping and reconstruction, the lesser the subsequent localization error in that environment. Localization vs. ground truth is compared for the two vs. three vehicle case in figure 4.6.

Table 4.2: Comparison of localization accuracy with respect to number of vehicles involved in NBMV

Number of vehicles	Map Density (pts)	MSE (cm)
Initial	147	59.8
2	344	25.5
3	552	17.4
4	621	15.6
5	654	14.8

At the same time, the NBMV planner is expected to balance this expected improvement with the additional path cost expenditure, so when the path cost parameter δ is higher, the NBMV planner can choose to use only some vehicles for mapping purposes. An interesting

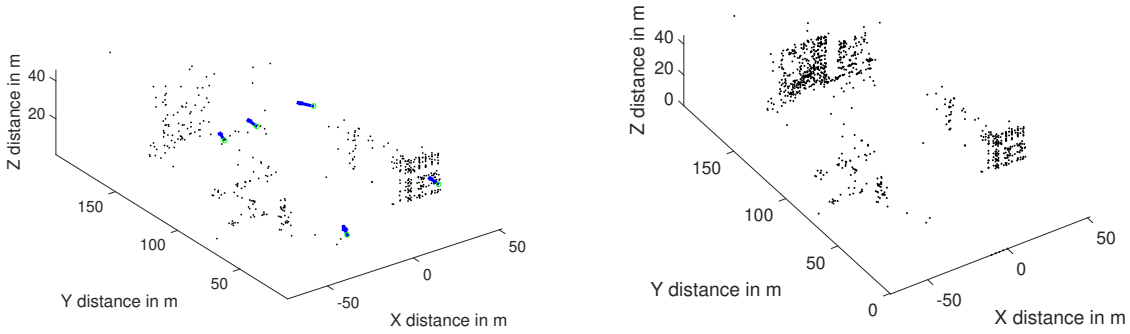


Figure 4.7: Result of the NBMV planner in an environment with five MAVs. Initial map and solution on left, final reconstruction on right. Reprinted from Vemprala and Saripalli[66]

Figure 4.8: Results of NBMV planner applied to a bigger environment. Initial map on left, final map on right. Middle shows original structure. Reprinted from Vemprala and Saripalli[66]

effect of using more than 2 vehicles can be seen in figure 4.7. Here, a large scene was asked to be observed by a set of five MAVs, but with a higher cost constraint. As the path cost is a sum of distances traveled by all the MAVs, the CMA-ES algorithm opted to return a solution with only three MAVs selected for examining the buildings closely, whereas the other two remained close to the starting location. It can be recalled here that the cost function was constructed in such a way that a minimum of two MAVs are required for examination to facilitate a successful map reconstruction, hence, three is a valid number of MAVs to be selected to capture images for an improved map. The initial and improved maps are shown in figure 4.7.

A few more scenes where the NBMV planner was applied can be seen through images

Figure 4.9: More NBMV examples: Images from initial poses (top) and images from optimized poses (bottom). Reprinted from Vemprala and Saripalli[66]

captured from initial and final positions of MAVs in figures 4.8 and 4.9.

4.4.2 VA-RRT* planning for individual MAVs

Under the assumption that a sufficiently informative map is already available, the VA-RRT* algorithm is meant to be executed by individual vehicles in order to result in vision-aware paths. Initially, the general characteristics of the VA-RRT* and performance are demonstrated through a set of simple experiments.

A typical result from the VA-RRT* shown figure 4.10 demonstrates its general operation. In this experiment, a synthetic 'map' of 3D points was initialized in the beginning to simulate a point cloud a vehicle might have access to, shown as a rectangular array of blue points in the figure. The VA-RRT* was asked to plan a path from the start location $q_{start} = (0, 0, 0)$ to $q_{new} = (100, 0, 0)$. The resultant path is shown in red, with camera views drawn at the nodes throughout the path. From the result, it can be understood that the VA-RRT* was able to successfully estimate the best views to be those which maximize feature distribution in the image plane and thus need to be in proximity to the map points. It can also be noticed that the VA-RRT* prioritizes spending more time in proximity to the map points, as obtaining multiple good measurements could result in a lower accumulated covariance that could be beneficial later in the path where the MAV is forced to move away from the points to reach its goal. It can also be seen how the entire tree, shown as light blue connections, is rewired in such a way that most edges lead to the area of good measurements. Under the

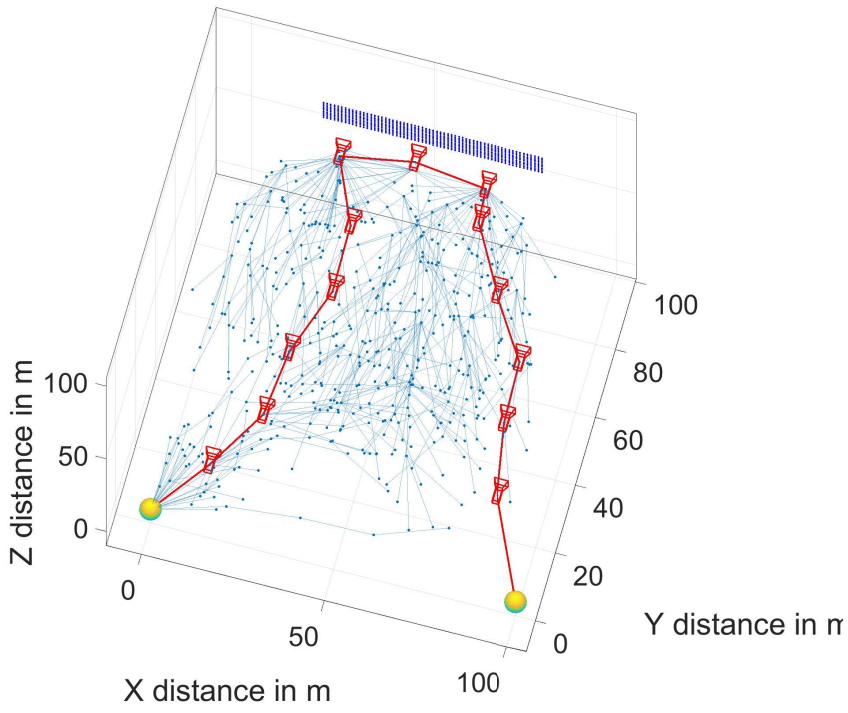


Figure 4.10: Sample plan from VA-RRT* that ensures good visual observations of features while moving from start to goal

assumption of static environments, this is beneficial for future planning queries.

Figure 4.11 shows sample paths returned by the VA-RRT* algorithm for different number of maximum iterations. This comparison shows the benefit of features such as vision-aware sampling and cascaded rewiring. The algorithm is able to plan paths that adhere to the idea of better feature data acquisition even after a low number of iterations, such as can be seen in figure 4.11.a: where the tree is already biased towards the area of good measurements. This can also be partly attributed to the Voronoi bias of RRT-variants.

The main parameters that control the performance of VA-RRT* are the weights within the cost function: w_c and w_Σ. The effect of different values of w_c and w_Σ can be best observed in the figures in 4.12. A high value for w_c results in a path that chooses low path cost over information gathering, and thus behaves similarly to algorithms such as the generic RRT*. Conversely, choosing a high w_Σ results in the algorithm prioritizing acquiring

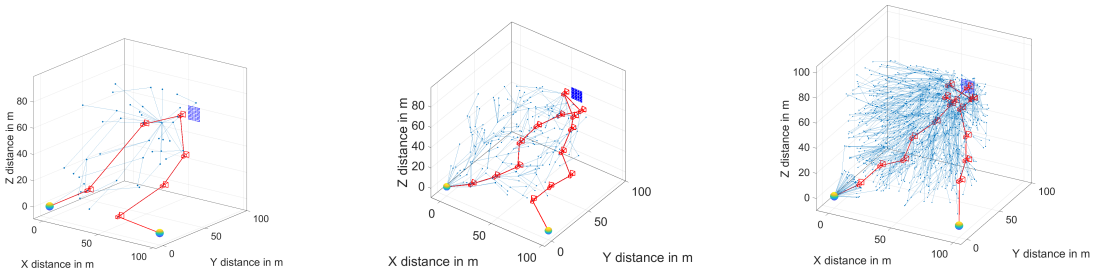
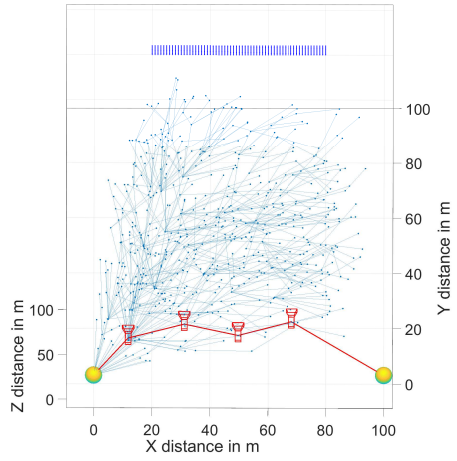
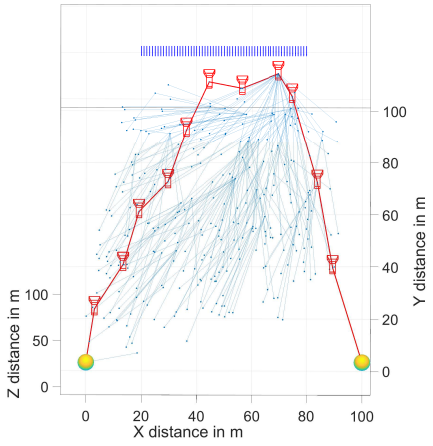


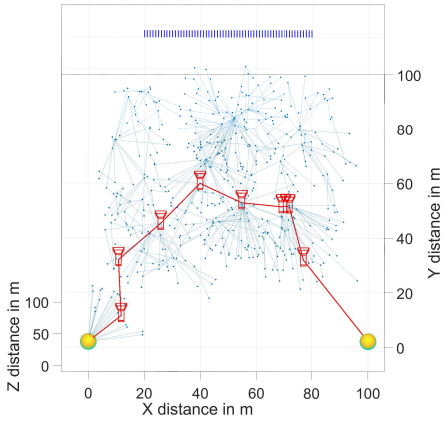
Figure 4.11: VA-RRT* expansion with number of nodes: 50, 100 and 1000 nodes



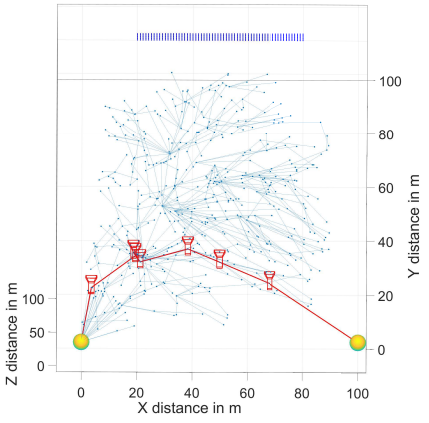
(a) High w_c, low w_Σ



(b) Low w_c, high w_Σ



(c) Balanced w_c and w_Σ



(d) Balanced w_c and w_Σ

Figure 4.12: Performance of VA-RRT* for different combinations of w_c and w_Σ.

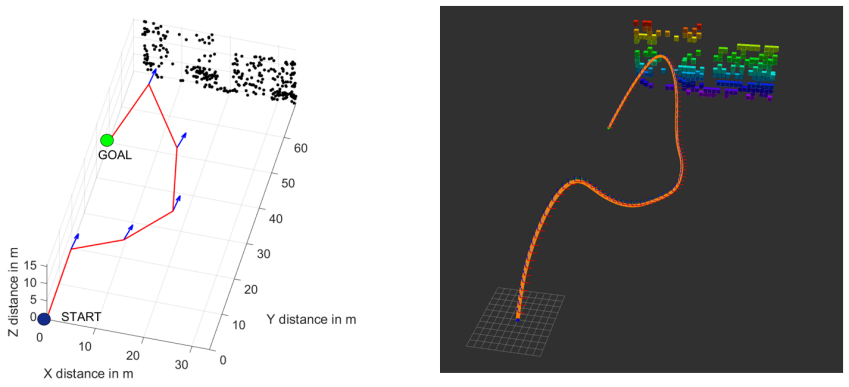


Figure 4.13: VA-RRT*: coarse, initial path plan on the left, minimum-snap optimized smooth plan on right.

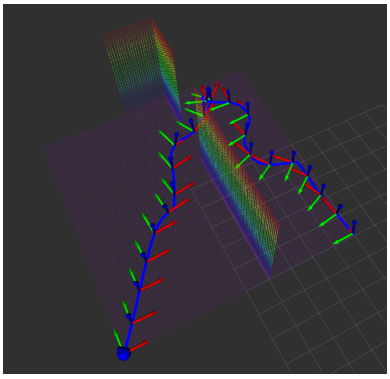
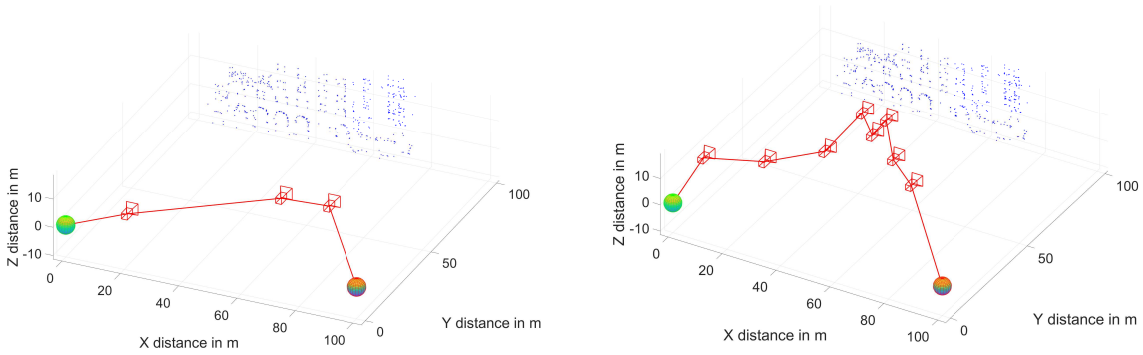


Figure 4.14: VA-RRT* demonstrating vision-aware behavior along with collision avoidance in a simulated room

better views of the map at the expense of path cost. It is possible to devise a balanced combination of these two values, which can result in paths such as the ones shown in 4.12.c and 4.12.d, where VA-RRT* attempts to optimize both path cost and uncertainty reduction.

The extension of VA-RRT* for minimum-snap trajectory generation results in feasible, smooth trajectories for MAV navigation. Figure 4.13 demonstrates this: given a map of points, the core of VA-RRT* is only responsible for a coarse connection of nodes (4.13 - left) indicating where the localization would improve. As the next step, the minimum snap trajectory generation takes this coarse connection and converts that into a smooth trajectory (4.13 - right). Figure 4.14 shows a more complex example where an environment resembling



(a) Balanced choice of w_c and w_Σ **(b)** High value of w_Σ

Figure 4.15: Path plans generated from the VA-RRT* for a sample map from AirSim.

two rooms and a doorway was synthesized, with only some of the walls containing texture (seen in the occupancy grid as colored points), and the others being blank, and thus of no use for localization. The VA-RRT* maintains view of a subset of these features throughout the trajectory from start to goal, and carefully plans the yaw angles (green arrow on axes) when moving through the doorway that allows continuous feature visibility. The trajectory is also planned such that the vehicle stays away from any chance of collision with the walls.

After validating the basic characteristics of the VA-RRT*, its ability to improve localization was tested using AirSim. Results from a sample experiment are shown here, where a 3D map was reconstructed from two MAVs (from viewpoints provided by the NBMV planner). A third MAV was placed at approximately 100m away from the map, and the VA-RRT* algorithm was asked to plan a vision-aware path from the start location to a goal location 100m to its left. Intuitively, the result from a shortest path algorithm would involve flying to the left in a straight line till the goal location is reached. On the other hand, VA-RRT* prioritizes obtaining better measurements and enhanced feature viewing, hence the paths planned by VA-RRT* are as shown in figure 4.15. Figure 4.15 shows two paths planned with different weighting parameters on the cost: increasing the weight on path cost results in a path that tries to balance both distance traveled and covariance reduction, similar to examples above.

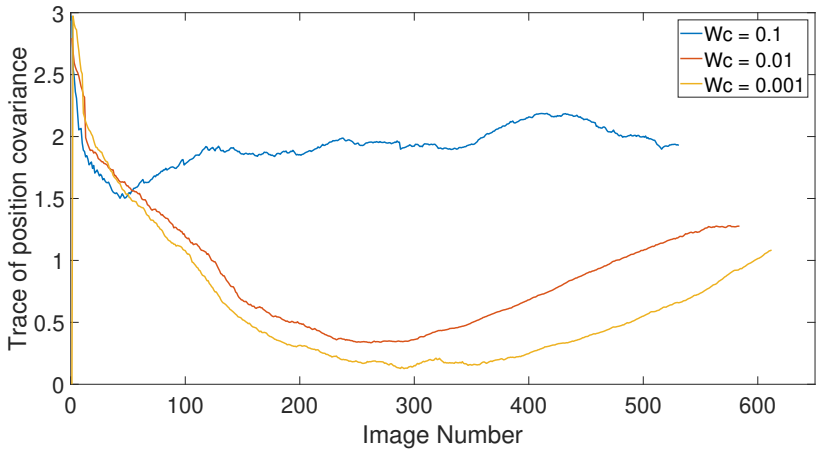


Figure 4.16: Comparison of covariance (trace) between VA-RRT* paths generated with different weighting parameters

Results from the localization algorithm prove that the VA-RRT* was indeed successful in reducing pose uncertainty. The trajectory that was estimated to reduce covariance is seen to actually reduce pose covariance as evidenced by the comparison of pose covariance traces in figure 4.16. While a path with high path-cost weight (close to a shortest path plan) resulted in high, increasing covariance, both the trajectories shown in figure 4.15 result in a decrease in covariance between start and goal.

4.5 Interleaved VA-RRT* and NBMV operation

The primary function of the VA-RRT* algorithm is to compute trajectories that are more optimal than a naive trajectory or one that only optimizes path cost, in the sense that feature based localization is always possible, and it can be improved wherever possible. From this statement, and some of the demonstrations shown in section 4.4.2, a connection can be made between this planning framework and the localization framework that VA-RRT*, whenever possible, attempts to reduce the need for inter-MAV localization. Inter-MAV localization is mainly beneficial when one of the MAVs is, for instance, suffering from bad feature tracking that results in an increasing pose uncertainty, or is just navigating at a farther distance from the map. Allowing VA-RRT* to act on these scenarios could result in a certain degree of improvement, where the planner could ask the MAV to be controlled so

that it is in view of more features, or just accumulate good measurements before moving to an area of the map where feature tracking may not work very well. In essence, using the VA-RRT* can result in less requirement of communication between the vehicles.

One scenario where VA-RRT* cannot perform to its expected level of uncertainty reduction is when the operation is deliberately constrained. In applications where energy expenditure needs to be minimal, or in a case where the MAV's battery level is too low to execute long maneuvers that could result in better localization, the VA-RRT* algorithm can be constrained by increasing the weight on the path cost parameter (w_c): and in this case, the VA-RRT* plans would have to make compromises in their expected covariance reduction. To handle these corner cases, it is possible to combine VA-RRT* and the previously described NBMV planner, to hand off the responsibility of pose uncertainty improvement to the inter-MAV localization module by identifying certain points in the path where communication would be necessary.

Figure 4.17 shows an example scenario where these two algorithms can be combined. Two MAVs were commanded to move away from a set of features but with a high value of w_c, a command that would result in reduction in feature distribution on the image plane. Consequently, the VA-RRT* has to plan an almost straight-line path from start to goal, while being aware that this would result in an insufficient decrease of estimated covariance - the covariance computed through information accumulation. The covariances expected throughout the path were analyzed, and if the least covariance expected in the path is not under a certain threshold (relative to the initial covariance), intermediate points in the path were marked to be candidates for improvement.

Once marked, a slice of the complete environment is isolated, with the points from the VA-RRT* path acting as 'seed' poses, and was passed to the NBMV planner. Naturally, the NBMV planner tries to maximize overlap and other relevant factors: and hence, it resulted in an improved pair of points - with the assumption that if inter-MAV localization were to be performed from these locations, it would result in enhanced accuracy. Allowing the NBMV

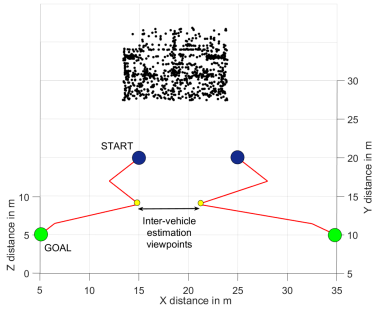


Figure 4.17: Interleaved VA-RRT* and NBMV operation. When VA-RRT* is not able to reduce covariance by expected amounts, NBMV can identify possible viewpoint sets from where inter-MAV localization can be attempted.

planner to only sample within a slice of the environment ensures that the new viewpoints selected by the NBMV planner do not require considerable energy expenditure.

5.1 Summary of contributions

In this dissertation, two algorithmic frameworks: one for collaborative localization and one for collaborative uncertainty-aware path planning were presented for vision-based micro aerial vehicles. The target application scenario involves a set of multirotor micro aerial vehicles, each equipped with a forward-facing monocular camera, as well as communication abilities with other vehicles. The collaborative localization is an approach that is solely vision-based, and uses a feature-based method. The localization starts with the vehicles sharing their feature data to build 3D maps of the surroundings. The VCL framework uses two types of localization: one which runs onboard the vehicles in a decentralized way, known as intra-MAV localization and another, that is performed by one vehicle for another: known as inter-MAV localization. Concepts from computer vision, multiple view geometry and structure from motion were combined to achieve collaborative mapping of environments, as well as the two types of localization mentioned previously. Different estimates coming from the other vehicles in the group are fused in a consistent manner using a technique known as covariance intersection, which does not require keeping track of cross-correlation terms. Mapping can be repeated over time to update the map point clouds with new features.

This collaborative localization was implemented as C++ software, and tested with image datasets from high fidelity simulation environments (Microsoft AirSim) as well as from real life MAVs in both indoor and outdoor scenes. Some of these experiments involved flights over large areas, some simulated flights well over 100m of distance, yet only exhibiting around position estimation errors in the order of 1-2% once the collaborative nature of the VCL algorithm came into play. In general, the VCL framework was shown to consistently result in good localization through data fusion when required. The findings show that

collaboration has the potential to improve localization in challenging environments and generally result in more accurate pose estimates, thus supporting the initial claim that collaboration can enhance localization accuracy. The implementation of the algorithm was designed in such a way that it does not require constant communication between the vehicles. Even when communication is required, such as in cases of inter-MAV localization, the data that needs to be transferred is only feature points and descriptors, generally in the order of kilobytes and thus can be handled comfortably by Wi-Fi networks. While this algorithm was not tested real-time in a feedback loop, a timing analysis of the modules involved shows that most of these computation tasks can be performed in the order of milliseconds by conventional hardware. There is potential for optimization of the existing code for better performance. With the proliferation of small, efficient GPU platforms such as the NVIDIA Jetson TX2 and Jetson Nano, the implementation of GPU-based feature detection and matching in VCL has the potential for fast, accurate localization onboard MAVs.

In chapter 4, the collaborative path planning framework was presented. This extends the idea of a collaborative MAV system into the planning phase, and attempts to adapt the same factors defined in the localization framework into an uncertainty-aware planning scenario. As for any feature-based localization framework, the major factors affecting localization quality were identified to be firstly the quality of reconstruction of the environment being used as the map and secondly, the quality of feature tracking from the given map. Proximity to feature-rich areas, fronto-parallel views, a proper baseline to depth ratio are some of the factors that influence the accuracy of reconstruction and subsequent localization. Hence, a set of heuristics were developed to describe these qualities mathematically, with the idea that the lower the values of these heuristics, the more improved these characteristics are. To improve mapping performance and to generate better 3D point clouds, a next-best-view planner was developed for multiple vehicles, called a next-best multi-view (NBMV) planner. The combination of multiple vision-specific heuristics was formulated as a cost function and

was adapted into an evolutionary algorithm called covariance matrix adaptation (CMA-ES), with the idea that solving this optimization problem results in a set of viewpoints the vehicles can move to, and then create better maps.

Alongside the next-best-view planner, which is mainly responsible for map improvement, a sampling based planner was also developed to take care of individual vehicle navigation and to ensure that feature tracking is always possible, and that the vehicles prioritize obtaining better observations of the feature map whenever possible. This planner, an RRT-based algorithm with features inspired by several recent variants of RRT, was named Vision-Aware RRT* (VA-RRT*). It is an extension to RRT*, the sampling based planner that exhibits asymptotic optimality characteristics: and contains a customized cost function that is a trade-off between minimization of path cost along with minimization of covariance with weighting parameters that allow the user to choose the priority. The expected covariance at various parts of the 3D environment is estimated through a heuristic-based measurement uncertainty value, which is eventually propagated through the trajectories by formulating good measurements as 'information gain'. Through several experiments in simulation, this planning framework was shown to be beneficial for localization. The NBMV planner was shown to consistently result in denser, more accurate point clouds, and it was demonstrated that these improved maps did also result in improved pose estimation accuracy. The VA-RRT* also exhibits similar characteristics, driving the vehicles efficiently towards texture-rich areas and maintaining observability of features throughout the trajectories. While the planning framework was only implemented in simulation, it was developed in a way that it is feasible to apply it on real MAVs. The VA-RRT* algorithm was coupled with a minimum-snap trajectory generation algorithm which ensures that any coarse high-level path returned by the RRT-based planner is converted into a smooth, quadrotor-feasible trajectory for navigation.

5.2 Possible applications

With the surging interest in the idea of micro aerial vehicle swarms, the constantly reducing size of vehicles, while enhancements are made in computational speed and communication, several possible applications can be identified with regards to using collaborative methods for autonomy. Considering the factors that are fundamental to the work presented in this dissertation, this section attempts to identify a few applications where the presented approaches would be useful.

5.2.1 Decoupled aerial stereo

The first application where collaborative localization and planning could be beneficial would be mapping of large structures such as buildings or natural structures. The classical structure from motion pipeline, which involves identifying sveral viewpoints and having a camera move between them while taking pictures; could be made more efficient by using a set of (possibly cheap, expendable) quadrotors. Proper imaging would require accurate positioning of the cameras and in situations where the structures might be so remote that GPS coverage is not guaranteed, a GPS-denied approach such as this would be beneficial. The NBMV planner, for example, could attempt to identify the best viewpoints for imaging the structure, whereas the collaborative localization can make sure that each MAV is at its position by considering the relative positions between the vehicles. While performing this navigation and imaging, the VA-RRT* would act as the high level planning framework ensuring localization is not lost during operation. For a simple operation with only two vehicles, this system could essentially act as a decoupled, variable-baseline stereo camera, thus the name decoupled aerial stereo.

5.2.2 Cooperative assembly

Another application where collaborative localization and trajectory planning could be highly useful is where multiple MAVs such as quadrotors work together to carry payload and to assemble larger objects. These applications usually involve two or more small quadrotors

attempting to pick up large objects by attaching themselves to the edges or the corners of the object, and thus, require very precise positioning. Intuitively, this can be seen as a sort of formation control, and hence a robust combination of individual and relative estimation is an attractive choice for positioning the vehicles in their precise spots and maintaining them through flight and payload carriage. At the same time, it would be desirable for the MAVs to navigate only through well lit areas, and to keep texture-rich areas in view as much as possible if vision is their primary sensing modality, hence the planning framework would be able to handle trajectory generation for safe navigation and payload transport.

5.3 Future Work

There are numerous possibilities of extending this current work. Adapting the current software into a real-time framework, capable of running onboard separate vehicles with communication capabilities would be a desirable extension. This deployment could also investigate the ability to handle communication delays, creation of a feedback control loop.

The algorithms used in the VCL framework could also be extended to match a more robust SLAM framework. Primarily, generating and including map uncertainty could assist in a higher degree of informativeness for localization and subsequent map improvement. It is also possible to use a more accurate system model within the Kalman filter to match the MAV dynamics. Another direction of work could involve integrating IMU measurements into the vision based framework in order to relax the assumption of knowing the initial estimate of scale: utilizing visual-inertial data would allow for online scale estimation and propagation.

REFERENCES

[1] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[2] C. Galiński and R. Zbikowski, "Some problems of micro air vehicles development," 2007.

[3] "Vicon motion capture system." `https://www.vicon.com/motion-capture/engineering`. Accessed: 2010-09-30.

[4] N. Abdelkrim, N. Aouf, A. Tsourdos, and B. White, "Robust nonlinear filtering for ins/gps uav localization," in *2008 16th Mediterranean Conference on Control and Automation*, pp. 695–702, IEEE, 2008.

[5] B. Yun, K. Peng, and B. M. Chen, "Enhancement of gps signals for automatic control of a uav helicopter system," in *2007 IEEE International Conference on Control and Automation*, pp. 1185–1189, IEEE, 2007.

[6] T. Templeton, D. H. Shim, C. Geyer, and S. S. Sastry, "Autonomous vision-based landing and terrain mapping using an mpc-controlled unmanned rotorcraft," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1349–1356, IEEE, 2007.

[7] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and slam for medium and high altitude flying uavs," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 137–161, 2009.

[8] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3, pp. 2799–2804, IEEE, 2002.

[9] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[10] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *2010 IEEE International Conference on Robotics and Automation*, pp. 1642–1648, IEEE, 2010.

[11] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *2011 IEEE International Conference on Robotics and Automation*, pp. 763–770, IEEE, 2011.

[12] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[13] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4972–4978, IEEE, 2012.

[14] J. Stowers, M. Hayes, and A. Bainbridge-Smith, "Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor," in *2011 IEEE International Conference on Mechatronics*, pp. 358–362, IEEE, 2011.

[15] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *2012 IEEE international conference on robotics and automation*, pp. 9–15, IEEE, 2012.

[16] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.

[17] 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 *2011 IEEE International Conference on Robotics and Automation*, pp. 3056–3063, IEEE, 2011.

[18] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for

indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[19] Z. Fang and S. Scherer, "Real-time onboard 6dof localization of an indoor mav in degraded visual environments using a rgb-d camera," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5253–5259, IEEE, 2015.

[20] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, IEEE Computer Society, 2007.

[21] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 15–22, IEEE, 2014.

[22] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[23] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, pp. 834–849, Springer, 2014.

[24] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *2010 IEEE International Conference on Robotics and Automation*, pp. 21–28, IEEE, 2010.

[25] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1872–1878, Sep. 2015.

[26] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572, IEEE, 2007.

[27] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[28] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[29] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1477–1484, 2011.

[30] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[31] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2815–2821, IEEE, 2012.

[32] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2609–2616, IEEE, 2014.

[33] A. Martinelli, F. Pont, and R. Siegwart, "Multi-robot localization using relative observations," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2797–2802, IEEE, 2005.

[34] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 1402–1409, IEEE, 2009.

[35] L. C. Carrillo-Arce, E. D. Nerurkar, J. L. Gordillo, and S. I. Roumeliotis, "Decentralized multi-robot cooperative localization using covariance intersection," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1412–1417, IEEE, 2013.

[36] J. Knuth and P. Barooah, "Distributed collaborative localization of multiple vehicles from relative pose measurements," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pp. 314–321, IEEE, 2009.

[37] V. Indelman, E. Nelson, N. Michael, and F. Dellaert, "Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 593–600, IEEE, 2014.

[38] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein, "Distributed vision-aided cooperative localization and navigation based on three-view geometry," *Robotics and Autonomous Systems*, vol. 60, no. 6, pp. 822–840, 2012.

[39] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 2, pp. 354–366, 2013.

[40] M. W. Achtelik, S. Weiss, M. Chli, F. Dellaerty, and R. Siegwart, "Collaborative stereo," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 2242–2248, IEEE, 2011.

[41] N. Piasco, J. Marzat, and M. Sanfourche, "Collaborative localization and formation flying using distributed stereo-vision," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 1202–1207, IEEE, 2016.

[42] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular slam with multiple micro aerial vehicles," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 3962–3970, IEEE, 2013.

[43] P. Schmuck and M. Chli, "Multi-uav collaborative monocular slam," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3863–3870, IEEE, 2017.

[44] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[45] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[46] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[47] S. Karaman, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104.

[48] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear pomdps by factoring the covariance," in *Robotics Research*, pp. 293–305, Springer, 2010.

[49] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 723–730, IEEE, 2011.

[50] D. Levine, B. Luders, and J. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," in *AIAA Infotech Aerospace 2010*, p. 3360.

[51] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[52] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[53] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[54] R. He, S. Prentice, and N. Roy, “Planning in information space for a quadrotor helicopter in a gps-denied environment,” in *2008 IEEE International Conference on Robotics and Automation*, pp. 1814–1820, IEEE, 2008.

[55] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, “Motion-and uncertainty-aware path planning for micro aerial vehicles,” *Journal of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

[56] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, “Perception-aware path planning,” *arXiv preprint arXiv:1605.04151*, 2016.

[57] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, “Receding horizon" next-best-view" planner for 3d exploration,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 1462–1468, IEEE, 2016.

[58] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “Pampc: Perception-aware model predictive control for quadrotors,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, IEEE, 2018.

[59] D. Silver, “Cooperative pathfinding.,” 2005.

[60] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, “Multi-robot collision avoidance with localization uncertainty,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 147–154, International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[61] F. Belkhouche and T. Jin, “An approach for collaborative path planning in multi-robot systems,” in *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2356–2361, IEEE, 2009.

[62] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[63] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of aerial robots," in *Robotics: Science and Systems*, 2013.

[64] S. Vemprala and S. Saripalli, "Monocular vision based collaborative localization for micro aerial vehicle swarms," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 315–323, June 2018.

[65] S. Vemprala and S. Saripalli, "Vision based collaborative localization for multirotor vehicles," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1653–1658, Oct 2016.

[66] S. Vemprala and S. Saripalli, "Vision based collaborative path planning for micro aerial vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, May 2018.

[67] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[68] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[69] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*, pp. 430–443, Springer, 2006.

[70] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," 2011.

[71] P. F. Alcantarilla and T. Solutions, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Patt. Anal. Mach. Intell*, vol. 34, no. 7, pp. 1281–1298, 2011.

[72] G. Levi and T. Hassner, "Latch: learned arrangements of three patch codes," in *2016 IEEE winter conference on applications of computer vision (WACV)*, pp. 1–9, IEEE, 2016.

[73] C. Parker, M. Daiter, K. Omar, G. Levi, and T. Hassner, "The cuda latch binary descriptor: because sometimes faster means better," in *European Conference on Computer Vision*, pp. 685–697, Springer, 2016.

[74] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 0756–777, 2004.

[75] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, p. 133, 1981.

[76] L. Moisan and B. Stival, "A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix," *International Journal of Computer Vision*, vol. 57, no. 3, pp. 201–218, 2004.

[77] L. Moisan, P. Moulon, and P. Monasse, "Automatic homographic registration of a pair of images, with a contrario elimination of outliers," *Image Processing On Line*, vol. 2, pp. 56–73, 2012.

[78] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[79] T. Ke and S. I. Roumeliotis, "An efficient algebraic solution to the perspective-three-point problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7225–7233, 2017.

[80] W. H. Press, *Numerical recipes in C*, vol. 2.

[81] G. Chang, "Robust kalman filtering based on mahalanobis distance as outlier judging criterion," *Journal of Geodesy*, vol. 88, no. 4, pp. 391–401, 2014.

[82] F. M. Mirzaei and S. I. Roumeliotis, "A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 1143–1156, 2008.

[83] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, vol. 4, pp. 2369–2373, IEEE, 1997.

[84] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[85] P. Moulon, P. Monasse, R. Marlet, and Others, "Openmvg." `https://github.com/openMVG/openMVG`.

[86] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`.

[87] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[88] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.

[89] C. Connolly, "The determination of next best views," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 432–435, IEEE, 1985.

[90] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proceedings of IEEE international conference on evolutionary computation*, pp. 312–317, IEEE, 1996.

[91] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*, pp. 649–666, Springer, 2016.

[92] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system,"

[93] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[94] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866, May 2012.

APPENDIX A

CAMERA PINHOLE MODEL AND PROJECTIVE GEOMETRY

A.1 Pinhole Model

A camera is a device that is capable of recording a three-dimensional object or a scene as a two-dimensional image. A simple model that represents the operation of a camera is what is known as the 'pinhole camera model', or the central projection model. In this model, a camera system is designed as a photographic sensor that is placed behind a barrier with a tiny hole in it, known as the aperture. It is expected that the 3D object that needs to be imaged is emitting multiple rays of light outward. Because of the small size of the aperture, only very few of these emitted rays reach the sensor, thus allowing a one-to-one mapping between the recording and the actual object.

The aperture is referred to as the pinhole, or the center of the camera. The recorded image is represented by a 2D plane, also known as the retinal plane. The distance between the retinal plane and the camera center is the focal length of the camera, denoted by f. To unify these two references, a coordinate system can be defined with the pinhole O at the center, called the camera coordinate system. Contrary to other common navigational coordinate systems like NED, for a system $[i \quad j \quad k]$, it is conventional in a camera coordinate system to have k pointing perpendicularly outwards from the pinhole.

Let $X = [x \quad y \quad z]^T$ be a 3D point being imaged by the camera. X, when visible to the pinhole camera, will be **projected** onto the image plane Π, resulting in a corresponding projection $X' = [x' \quad y']^T$. It can be gathered from image A.1 that, through the law of similar triangles between OCX' and OXZ,

$$X' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \end{bmatrix} \tag{A.1}$$

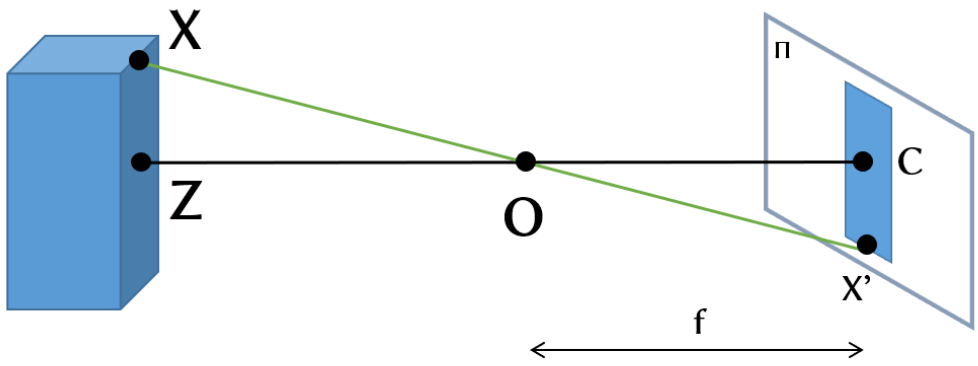


Figure A.1: Pinhole camera model

This $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ mapping is known as the projective transformation. While the consideration so far has been that of a perfect image plane, in reality, digital images introduce several of their own transformations, such as distortion. One basic transformation that needs to be appended to the one in equation A.1 is that in an image, the origin is usually at the top-left corner, instead of the center as in the image plane. Hence, the transformation in A.1 can be modified as follows:

$$X' = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} + c_x \\ f\frac{y}{z} + c_y \end{bmatrix} \tag{A.2}$$

This nonlinear relationship between the object point and the image point can be re-expressed as a matrix product for simplification. This is usually achieved by introducing what is known as a homogeneous coordinate system. In this homogeneous coordinate system, a new unit coordinate is appended to each vector: thus changing the image coordinates to $X'_h = [x' \quad y' \quad 1]^T$ and the object coordinates to $X_h = [x \quad y \quad z \quad 1]^T$. Now, the same relationship in A.2 can be rewritten in the homogeneous system as

$$X'_h = \begin{bmatrix} fx + c_x z \\ fy + c_y z \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{A.3}$$

Dropping the h subscript and considering entirely homogeneous coordinates, this can be further decomposed as follows:

$$X' = \begin{bmatrix} fx + c_x z \\ fy + c_y z \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} X = \mathbf{K} \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} X \tag{A.4}$$

The matrix $\mathbf{K}$ contains the internal parameters of the camera such as focal length and principal point (and occasionally, skew parameters), and is called the camera matrix or the intrinsic matrix. The intrinsic parameters of a camera are usually computed through a calibration procedure that involves comparisons between known 3D points and their 2D projections. It is also usually necessary to estimate the distortion parameters of cameras, as distortion is very commonly introduced in images due to the usage of lenses.

The relationship in equation A.4 assumes a camera placed at the origin of the world coordinate system. If the camera were to be at an arbitrary position and orientation that can be captured by a translation vector $\mathbf{t}$ and a rotation matrix $\mathbf{R}$, the projective transformation is converted to

$$X' = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} X \tag{A.5}$$

This equation represents the complete mapping of a 3D point X in an arbitrary world reference frame to an image plane. The matrix $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$ is known as the extrinsic matrix, the extrinsics encoding the external parameters of the camera, i.e., pose. The function of many camera pose estimation algorithms, and especially the ones described in this dissertation are to estimate the extrinsic parameters given the intrinsic parameters, and the

point correspondences. The intrinsic and extrinsic parameters together form the camera projection matrix.

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{A.6}$$

APPENDIX B

COVARIANCE OF CAMERA POSE ESTIMATION THROUGH REPROJECTION ERROR

Typically, the minimization of reprojection error as described in section 3.6 is solved through a non-linear least squares formulation. Although pixel space errors are usually Cauchy distributed, it is conventional to adapt a Gaussian error model when considering camera parameters in the context of refinement. Hence, an additive error model of the form below can be constructed for the reprojection.

$$X_i = \pi(x_i, \theta) + u_i \tag{B.1}$$

The sum of squares of the reprojection error of each 3D-2D correspondence is to be minimized, with the sum of squares expressed as

$$S = \sum_i r_i^2 \tag{B.2}$$

where r_i represents the residual, typically written in standard form as follows:

$$r_i = \mathbf{X_i} - \pi(\mathbf{x_i}, \theta) \tag{B.3}$$

S is minimum when the gradient is zero. Thus, the minimization condition would be

$$\frac{\partial S}{\partial \theta_k} = 2\sum_i r_i \frac{\partial r_i}{\partial \theta_k} = 0 \tag{B.4}$$

It can be recalled that the initial estimate of θ is already available: either from the PNP algorithm or the 5-point algorithm in case of inter-MAV estimation. Hence, the function of the least squares is to minimize S through incremental changes of θ, where the model is

linearized at every iteration by approximating the Taylor series expansion.

$$\theta_{k+1} = \theta_k + \Delta\theta \tag{B.5}$$

$$\begin{aligned} \pi(\mathbf{x}_i, \theta^{k+1}) &= \pi(\mathbf{x}_i, \theta^k) + \sum_j \frac{\partial \pi(\mathbf{x}_i, \theta_k)}{\partial \theta_j}(\theta_j^{k+1} - \theta_j^k) \\ &= \pi(\mathbf{x}_i, \theta^k) + \sum_j \mathbf{J}_{ij}\Delta\theta \end{aligned} \tag{B.6}$$

The Jacobian $\mathbf{J}$ represents the change in reprojection error for changes in pose. Extending the expression in B.5 results in the well known normal equations for non-linear least squares, written as

$$(\mathbf{J}^T\mathbf{J})\Delta\theta = \mathbf{J}^T\Delta\mathbf{X} \tag{B.7}$$

For any estimate $\hat{\theta}$, the covariance matrix of the estimator can be written as

$$\mathbf{\Sigma}_{\hat{\theta}} = E[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T] \tag{B.8}$$

For one iteration round that resulted in a value θ, the current value of θ would be

$$\hat{\theta}^k = (\mathbf{J}^{k^T}\mathbf{J}^k)^{-1}\mathbf{J}^k\mathbf{X}^k \tag{B.9}$$

Therefore, the covariance can be rewritten as

$$\begin{aligned} \mathbf{\Sigma}_{\hat{\theta}} &= E[((\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T u)((\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T u)^T] \\ &= (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T E[uu^T]\mathbf{J}(\mathbf{J}^T\mathbf{J})^{-1} \\ &= (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\sigma^2\mathbf{J}(\mathbf{J}^T\mathbf{J})^{-1} \\ &= \sigma^2(\mathbf{J}^T\mathbf{J})^{-1} \end{aligned} \tag{B.10}$$

Under the assumption that the Gaussian noise parameters are drawn from a zero-mean, unit-variance multivariate Gaussian distribution, the noise covariance would be equal to

the identity, simplifying B.10 to

$$\mathbf{\Sigma}_{\hat{\theta}} = (\mathbf{J}^T\mathbf{J})^{-1} \tag{B.11}$$