Autonomous Flight and Navigation in Air-Ground Systems

By
Benjamin Holden

Thesis submitted to the Faculty of
Carnegie Mellon University in partial fulfillment
of the requirements for the degree of
Master of Science
In
Robotics

APPROVED:

Maxim Likhachev, Advisor
Faculty Committee Member
maxim@cs.cmu.edu

Michael Kaess
Faculty Committee Member
kaess@cmu.edu

Venkatraman Narayanan
RI PhD Committee Member
venkatrn@andrew.cmu.edu

July 15th, 2016
Pittsburgh, Pennsylvania

Technical Report Number: CMU-RI-TR-16-42

Abstract

Robots are being used in situations that require more robust navigation and search. Often in these areas GPS can be unreliable or not present at all. In these situations, it becomes viable and even beneficial to consider collaboration between heterogeneous robots in order to utilize the different strengths of these robots, while avoiding situations that a specific robot's weakness would decrease the probability of a successful mission. A specific example of this is a scenario involves unmanned ground and air vehicles (UGV, UAV), where the ground robot contributes its high payload capacity to provide computational resources, large battery life, and high accuracy sensors to provide high accuracy localization while the aerial robot can bring its high mobility to explore hard to get to regions. In these situations, the UAV can have different levels of sensing capability. UAVs with poor sensing capabilities need a way to navigate and explore different environments. For this problem a state lattice planner augmented with controller-based motion primitives (SLC) is used to provide UAVs with a robust way to navigate while utilizing a large variety of sensors and even the help of other robots in the environment.

Another type of UAV has good sensing capabilities and is able to localize very well in x and y. However, the method for height estimation can often be naïve, especially when LIDAR data is readily available. This method is sensitive to surfaces with debris or objects on the ground plane, especially ramps, which can then lead to a poor height estimation. To solve this problem for the Search-Based Planning Lab (SBPL) my approach is to use these LIDAR scans to build a probabilistic elevation map (PEM) of the environment, which is then used to better estimate the height of the UAV as it traverses over these objects and environments. This method is also shown to work with dynamic obstacles and can be used to provide more features for the UAV.

Table of Contents

Acronyms

AR – Augmented Reality
SLC – State Lattice with Controller-Based Motion Primitives
MR-SLC – Multi-Robot SLC
PEM – Probabilistic Elevation Map
UAV – Unmanned Aerial Vehicle
UGV – Unmanned Ground Vehicle
SLAM – Simultaneous Localization and Mapping
LIDAR – Light Detection and Ranging
SBPL – Search-Based Planning Lab
PAD – Planning with Adaptive Dimensionality

List of Figures

List of Tables

1. Introduction

Autonomous flight and navigation in air-ground systems is a large field of study which has a lot of room for improvements in terms of the algorithms and systems used. It is also a field that is rapidly expanding with the design of new state of the art algorithms all of the time. Two different air-ground systems are explored in this work. In both cases the UGV has good sensing capabilities and endurance. The distinction comes in the first case where the UAV has poor localization capabilities and the second case is where the UAV has good localization capabilities. The first case of poor localization is discussed first in this work in the context to the multi-robot state lattice with controller-based motion primitives (MR-SLC) experiments. In this work I was responsible for the software and some of the electrical and mechanical engineering of the various motion controllers which were used to navigate the UAV from its starting location to its end goal. The other situation is one where the UAV has good localization capabilities and does not need or have localization assistance from the UGV. In this situation we found that it is important to have a robust method for estimating the height of the UAV in arbitrary environments with dynamic obstacles. In this work I will argue that this problem of state estimation does not have a universal solution but, that given a panning LIDAR system it is very beneficial to build a probabilistic elevation map (PEM) in order to estimate the height of the UAV as it traverses an environment. Both of these works are useful contributions to the field of air-ground robotic systems and not only helped their parent projects succeed, but provide a solid framework for future experiments and research questions based off them.

1.1 Overview of Motion Control and State Estimation

The main focus of this work is on motion control methods and state estimation in air-ground systems. Motion control is the means by which a robot navigates through an arbitrary environment. Most modern motion control approaches focus on metric motions or a different motion controller and these different motion controllers lack the optimality and utility of state lattice with controller-based motion primitives (SLC). This work allows for multiple motion controllers to be utilized by a planner in such a way as to minimize the cost of the path of the robot. The other focal point of this work is on state estimation in air-ground systems, which is a huge problem for highly dynamic robots such as UAVs. State estimation is how a robot decides where in the environment it is currently at and can include a large number of degrees of freedom (DOF). Most UAVs have 6 DOF being x, y, z, roll, pitch, and yaw and in this work I will focus on a method designed for accurately (and quickly) estimating the height of a UAV in a probabilistic manner for SBPL's lab Hexacopter. This means that the method can provide with high certainty a height estimation that is very fast, accurate, and subsequently allows for more stable control of the system on a whole.

1.2 Challenges

In this work we attempted and succeeded at finding solutions to various current problems in the field of air-ground robotic systems. Both of these challenges are very large and widely studied with many existing solutions depending on the needs of the system and the topology of the environment. The first challenge is to be able to navigate an area or areas in which localization is minimal or nonexistent. This problem is very difficult because without localization the robot will become lost and fail to reach its goal. The way we tackled this problem was to use a set of controller-based motion primitives, which can be used in localization denied areas to utilize additional sensors and still provide a means by which a robot can navigate through an environment. The other challenge is a sub-problem of the large problem of state estimation of a highly dynamic UAV. In this work we focused on a method of better estimating the height of a UAV given a 2D panning LIDAR. This problem has many solutions, which are great solutions in various situations, but none fit the practical approach using 2D LIDAR data that we desired. In addition, many any other modern solutions still struggle with the problem of dynamic obstacles in the environment and our work attempts to tackle this problem by containing a probabilistic element in our solution.

1.3 Scope

This work details the desire and need to have a solution to two different problems in the field of air-ground robotic systems. These problems are in areas that are continuously growing and as a result there is much work being done that is closely related to what is discussed as our solutions to these problems. The next section is on the controllers for the air-ground system, in which I detail the different work I did on motion controllers for a low cost UAV. This section also includes an experimental analysis of the work, which details the performance of the system on a whole. Height estimation is the subsequent section and details the work I have done on a new method for height estimation of a UAV by utilizing a 2D vertical LIDAR to build a probabilistic elevation map. This section also compares the performance of this with the naïve approach with a LIDAR used to estimate height. After these sections my contributions on a whole are summarized and the work is concluded.

2. Related Work

In the field of Mobile Robotics, state estimation and motion control have always been big problems, which are in constant need of improvements to the state of the art algorithms and methods. As the hardware in robots gets faster, sensors get better, and new algorithms are developed, robots become capable of much more impressive tasks. This improvement is very crucial to the field of robotics because the system as a whole is a very complex one where all the different components have to work together to determine the performance and if one part is significantly weaker then every other parts, then the system will suffer and not perform as well as it could have. That being said it is important to note that different algorithms work better for different systems and different topologies and an algorithm that may be dominant in one setup may perform worse than an alternate one in a different setup. This means that there is currently no universal solution to these problems at the moment.

2.1 Motion Control

Motion control is very important because it provides the actual control of the robot to allow the robot to accomplish different tasks. For a ground robot that maybe navigating around obstacles to get to a goal location or for a manipulator it may be picking up an object that needs to be sorted and put away. Motion control is a very large field that covers a large range of different subtopics. In this paper the motion control is largely based off a recent work in creating different controller-based motion primitives and as a result of this being a relatively new approach the work already done in fields similar to air-ground systems is relatively small.

State lattice with controller-based motion primitives (SLC) was first introduced by Butzke et al. in 2014 [1]. In its inception this framework was used to augment lattice planners with additional planning options that would take advantage of robots that had poor sensors or environments with poor GPS signal and other localization restrictive settings. This allowed for planning to still take place in these situations and get from one location to another by utilizing different sensors. A simple GPS denied example from this work is shown below in Figure 1 and shows how the lattice planner can be augmented with different motion primitives in order to generate feasible plans.

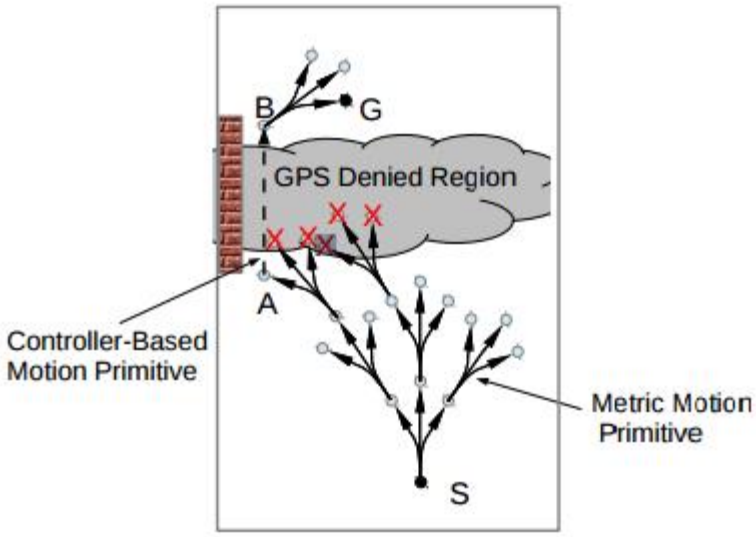

Figure 1: SLC example with metric motion primitives augmented with wall following motion primitives in GPS denied areas

The above figure shows an environment with an agent that uses GPS to localize and navigate through a map of the environment. However, in this example there is a large GPS denied region in between the current state and the desired state of the robot. Without augmenting the lattice planner an optimal solution may come back as a series of metric motions that go around the GPS denied area, which would have a very high cost because of the long path the robot would have to take. In this example the robot has additional sensing capabilities, which allow it to follow a wall from the start to the end and this can be used to add in an additional motion primitive in the form of a wall following motion primitive. This motion primitive is then added to the lattice planner who now returns the optimal path shown in the figure above. By adding these additional motion primitives, the robot is able to navigate an area that was previously not navigable and was also able to generate a valid trajectory at a fraction of the total path cost.

Other people have come up with algorithms to try and solve the problem that SLC solves, but they are always lacking in at least some area of the problem. Different controllers were developed such as a wall following controller [2], go to goal sensing abilities using potential fields [3], and different navigation functions [4]. A corner locating and localization algorithm has also been proposed [5]. These approaches were good because they utilized additional sensing capabilities the robots had access to but fell short in that they did not minimize the cost of the path, were prone to getting trapped in local minima, or could be difficult to abstract for arbitrary environments with arbitrarily shaped robots [6] [7]. Unlike these SLC provides cost minimization and can be easily abstract to arbitrarily complex environments [1].

Work has also been done using sequential controllers rather than just one as used by the people in the above works. In this way the robot is able to move from one controller to the next as it navigates through the environment, which eliminates the need for a controller that will work in every situation and environment for a given robot. These specific controllers are called by the

planner to perform a navigation function before the next controller is called [8] [9] [10]. These works are similar to the SLC framework in that they utilize many controllers, but they still lack in their ability to utilize a large range of perception triggers such as following a wall until an opening is detected on the other side of the robot. Implementing perception into the planning process is not a new idea and was used in [11], but again lacked the cost minimization that SLC has. Instead of minimizing cost they try to switch between the different controllers when a lack of progress is detected [12].

2.2 State Estimation

State estimation is another huge area in the field of mobile robotics. In this work I will be exclusively talking about height estimation for a UAV with a panning LIDAR, but there are many different techniques and algorithms used for general state estimation. The simplest method for estimating state comes from Kalman filtering [13] different sensors such as the IMU, a range sensor, and an optimal flow sensor. This setup is often found on commercial UAVs and is the best method to use with the sensors listed above. The methods for state estimation get much better with the addition of the camera and LIDAR sensors to the robot.

LIDARs make it possible to perform Simultaneous Localization and Mapping (SLAM) on a robot. SLAM is extremely useful because it builds a map of the environment while matching sensor readings to these maps in order to determine the state of the robot. This is a great method because it uses probabilistic methods to try and match features detected by the sensor (LIDAR or more recently cameras) to the map of features the robot has stored away and by doing this drift in the estimation of the robot's position is basically eliminated in environments with useful features [14]. Popular methods of state estimation in UAVs often involve combining an IMU with a 2D LIDAR and getting the state from that data [15] [16] [17]. A drawback of LIDAR based methods is that the LIDAR is often only used for 2D SLAM to find the x and y of the robot and the other 4 degrees of freedom (DOF) of the robot are found without the help of the LIDAR. This is done because 6 DOF SLAM can be very computationally intensive and leaves a need for a robust way to estimate the height of a UAV in a featureless dynamic environment. The other popular method is to use a camera or a stereo camera to perform visual SLAM which is then combined with the IMU in order to estimate the state [18] [19] [20]. Both of these methods are very well documented and perform well in static environments. However, if the map is not static and lacks in features then the system may have some trouble estimating the full state. There are many different variations on the SLAM algorithm and some can be used quite effectively on smaller aerial vehicles [21] [22] [23].

Some very notable recent methods for localization and mapping incudes PTAM [24], KinectFusion [25], Orb-SLAM [26], ant LSD-SLAM [27]. These methods show a lot of promise for localizing and mapping on micro aerial vehicles and are all vision based methods. While these methods are so promising this work will present a practical system for a LIDAR based Hexacopter. This LIDAR based method revolves around building an occupancy grid-like elevation map in order

to match a height map to LIDAR scans. Work has been done in trying to build elevation maps of an area to be used at a later time for various purposes, but at the moment no one is using elevation maps to try and actively estimate the state of an UAV. The most common use of an elevation map is for terrain mapping and there has been work done on how to use a UGV to build a probabilistic elevation map (PEM) [28] [29]. Elevation maps have also been used for navigation purposes in UAVs, but still not used in active state estimation [30] [31]. The work presented later maintains an elevation map similar to the way that occupancy grids work [32] [33].

3. Controllers for Air-Ground SLC

In many situations metric motions may not be available to robots in an environment. For example, the lack of GPS or other forms of reliable localization may make it impossible to navigate an environment without the use of additional sensors. In these situations, it becomes very important to utilize the full sensing capabilities of the robot with the addition of controller-based motion primitives. In this section I will give a brief overview of the State Lattice with Controller-Based Motion Primitives framework (SLC), how it can be used, and how we used and implemented it in our Multi-Robot SLC experiments [34]. All of the controllers detailed in this section can be used to provide additional navigation capabilities to a UAV in a UAV/UGV system and only one of the sets of controller-based motion primitives require the UGV's assistance.

3.1 State Lattice with Controller-Based Motion Primitives

State Lattice with Controller-Based Motion Primitives is a framework first introduced by Butzke et al [1]. This work introduced a solution to the problem detailed above. This problem, of not being able to navigate an environment when only poor localization is available, was approached by adding in these new controller-based motion primitives to the normal state lattice framework. This allowed for the utilization of sensors, which were previously not being used to their full potential. An example of this would be wall following. Following a wall only requires that the agent starts next to the wall and has some way of tracking alongside it until there is an ending in the wall. Figure 2 (a) below shows an example scenario of a map and (b) shows the graph constructed using controller-based motion primitives.

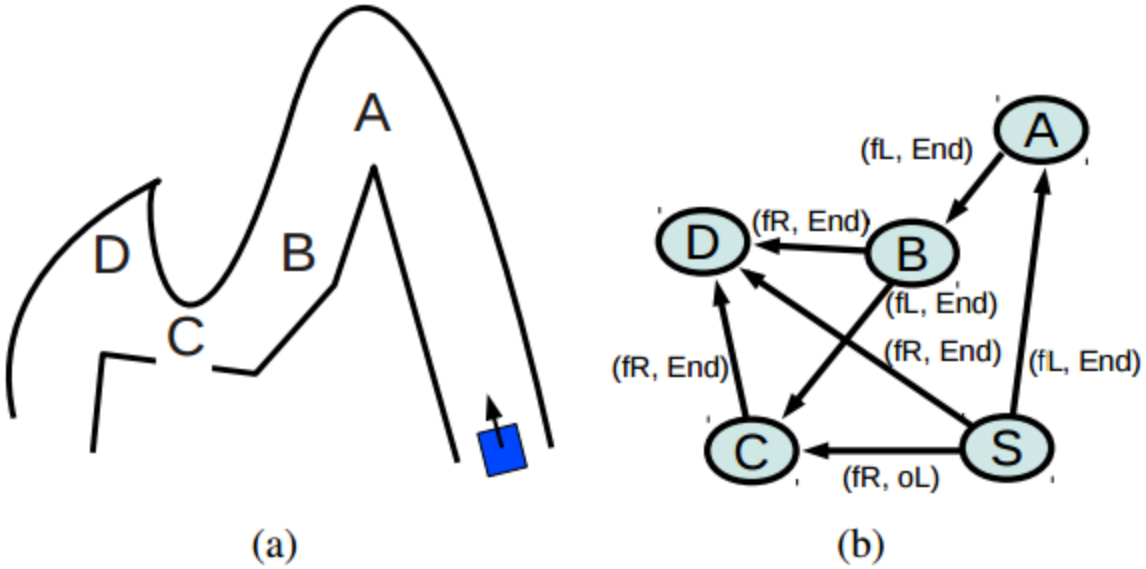

Figure 2: (a) Map of the environment and (b) a graph built using controller-based motion primitives

The important part of SLC comes from the labels on the edges in the graph above. Each edge is labeled with two different important characteristics of the motion primitive, the first of which shows which wall it is following. Wall following can either follow the wall to the left or the right of the robot. This symmetry is specific to this case but helps give an example of what different motion primitives can be. The second label is perhaps even more important to understanding SLC and that is the idea of end triggers. Since localization is not always reliable in areas that SLC is being used, it is important to have these triggers associated with each motion primitive, which represent the end condition of the primitive. For the wall following example there are a couple different end triggers that can be used, the first is simply that the wall has come to an end. The second is the wall opposite the one being followed comes to an end or opening. Both of these sets of triggers are useful to have and can be used to vastly increase the mobility of a robot in an indoor environment.

3.2 Controllers for a Low Cost Drone using MR-SLC

Using a low cost UAV has many advantages in terms of cost and mobility, but it can also introduce problems based off the UAVs weaknesses, such as poor localization capabilities. In the case of an UAV with poor localization we would like to both utilize the sensors available to the UAV and any other robots in the environment. Figure 3 below shows the UAV we used in our experiments. It is a Pixhawk based quadcopter with an ODroid attached through serial communication. It is outfitted with a forward facing camera and six IR point range finders with two on the left and right sides and one on the front and back sides.

Figure 3: Low Cost Pixhawk Based UAV with a camera and point range sensors

One such application of this idea is the Multi-Robot State Lattice with Controller-based motion primitives (MR-SLC). In this situation we would like to work with the UGV in order to get the UAV from its starting position to its goal position. In this case the UGV has excellent localization capabilities and the UAV can utilize this by localizing itself with respect to the UGV in the environment. Figure 4 below shows the UGV used in the MR-SLC experiments. It is a Segway based UGV with scanning LIDARs and a full computer stack on board. This UGV also has an AR bundle attached on top of it.

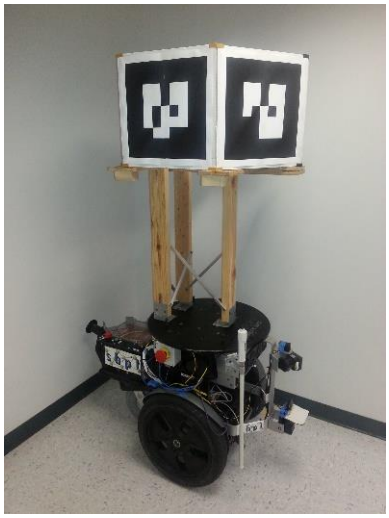

Figure 4: UGV with AR Bundle Attached

This means that using only a cheap camera on the UAV we can identify and localize off AR tags on the UGV. While this localization is available, metric motions become viable for the UAV and provide a means for robust navigation. However, the UGV cannot always be with the UAV in an environment and we would like to have more controller-based motion primitives that the UAV can use to navigate an environment even if it cannot accurately localize itself in the environment.

Some sensors that could be available to a UAV which it can utilize in this manner include cameras, LIDARs which are too sparse to use for SLAM, and point range finders. These sensors,

while not able to localize the UAV in the environment can be utilized to provide robust navigation in the form of controller-based motion primitives. Examples of these controller-based motion primitives include a wall following controller, a corner following controller, an AR assisted metric motion controller, and many more are both useful and feasible. All of the following controllers were implemented on a Pixhawk quadcopter with an ODroid running ROS attached.

3.2.1 Metric Motion Controller

The simplest of the motion controllers is a metric motion controller. This controller allows for the controlling of the UAV by inputting a desired change in state from its current state. So based off the current position of the UAV at {x, y, z, yaw} and input of {Δx, Δy, Δz, Δyaw} the UAV will attempt to move to {x + Δx, y + Δy, z + Δz, yaw + Δyaw}. All of these are in the map frame of the UAV so it becomes necessary to know the transform between the global map frame and the UAV's map frame.

Since the UAV cannot provide sufficient localization this transform between the two frames will likely drift over time and pure metric motions become an unreliable way to control the UAV. Because of this unreliability when planning with controller-based metric motions it is advisable to assign a high cost to plans that have pure metric motions in them and to only use them for small motions to transition between other controller-based motion primitives.

Because of these restrictions placed on the metric motion controller it becomes useful to break down the metric motion controller into a set of simple metric motions in order to reduce the possible complexity of the desired motion of the UAV. By reducing the complexity of the system, we can try and reduce the error introduced into the estimate of the UAV's position. In the MR-SLC experiments we broke down the metric motion controller into two simpler motion primitives. The first was a step controller which took in a distance and moved the UAV forward (body frame of the UAV) by that amount. The second of these was a yaw controller which took in a desired change in yaw and moved the UAV by that amount.

3.2.2 Wall Following Controller

The second of the motion controllers is the wall following controller. In order to use this controller, it is necessary to be able to take in the map of the environment and be able to detect walls as well as possible end triggers for them. The triggers for wall following include an opening on the right, an opening on the left, and a wall straight ahead. The wall following controller is, as its name suggests, motion primitives based off following a wall until an end condition is met. The UAV used in the MR-SLC experiments has two IR point range sensors on both the left and right sides of the UAV. This means that the wall following can be used to follow a wall on the left of the UAV or on the right of the UAV. Because of the dynamics of the UAV it is also able to follow the wall going forwards or going backwards (UAV body frame).

When planning paths, the plans can choose to follow a wall on the left or right traveling forward or backwards and can choose between following the left wall until it ends or following

the right wall until it ends. In order to follow a wall, the UAV needs to be able to maintain both distance and orientation from the wall and by having two IR sensors on either side this is possible. The way we did it was simple, take the average of the distance measured by the front side sensor and by the back side sensor and use that to find how much displacement is needed to get the UAV to maintain a set distance from the wall. The distance between the sensors and the difference between the measurements is also used to find the error in angle of the UAV in order to stay perpendicular to the wall.

Since the wall following motion primitive does not depend on the length of the wall a constant desired motion is given to the UAV in the direction parallel to the wall at every given point in time. This allows the UAV to be able to follow all walls regardless of their length and can even work on walls that are curved.

Detecting wall ending conditions is the other half of the wall following motion primitives. Walls can end in two different ways, the first way that a wall can end is that there is an opening on the side being tracked. This opening can be because of a corridor branching off, an open door or a few other environments. These types of opening are detected by the leading sensor switching from measurements to the wall to measurements that are very large indicating open space, while the rear sensor continues to detect wall measurements. When this occurs the wall is considered to be at an end and the UAV is commanded to hold position until it receives the next motion in the plan. The other way a wall can end is to reach an inside corner in a hallway or a room. An inside corner is when the wall comes to an end by connecting up with a perpendicular wall. This wall ending is detected by the IR sensor on the front or back of the UAV (depending on the direction of motion) going from detecting free space to detecting something in front of the UAV at or below a predetermined distance ahead. When this occurs the UAV stops and tries to hold position until the next command is received.

3.2.3 Corner Following Controller

Another set of controller-based motion primitives that are useful and even needed to navigate in some indoor scenarios come from the corner following controller. The corner following controller can be used to turn right around a corner or left around a corner. Similar to the wall following controller in the MR-SLC setup the topology of the UAV allows for navigation going forward or backwards around a corner on either the left side or the right side. Unlike the wall following motion primitives the corner controller motion primitives are created from a set of other controllers.

The nature of the corner controller makes it so that the corner controller should be used to transition from following one wall to following another wall. Because of the way it was designed for the MR-SLC experiments, the corner controller starts by checking the side IR sensors to make sure wall following is complete. This check acts as a safety check to make sure the robot does not turn into the wall and if the wall following is not fully complete it will travel forward until the wall following is registered as complete. At this point a step controller will take over and move the robot forward a predetermined distance, which is a function of the robot's size

and the size of the corridor. After the robot has stepped into the middle of the hallway or room's corner via the step controller the yaw controller will then take over and rotate the robot so that it is parallel to the new wall it will be following. Lastly the step controller is called again and it will drive the robot forward until a wall is detected, indicating that wall following can be used again.

At the start and the end of the corner controller motion primitives the robot is aware of its approximate location because of the nature of the environment, but in between relies solely on the metric motions of the robot, which means that it is important to tailor the step controller amounts to fit the specific robot's shape and size. A bad estimate could lead to the robot moving too far forward or not far enough. The likelihood of this potential problem can be reduced by utilizing the IR sensors during metric motions to try and avoid collisions and could even be used to fix the stepped amount if utilized correctly.

3.2.4 Augmented Reality Assisted Metric Motion Controller

A variant of the metric motion controller is the Augmented Reality (AR) Assisted Metric Motion Controller, which allows for the use of metric motions by temporarily providing the robot with the means of localizing itself in the environment. This is done by artificially placing AR tags in an environment that the robot will be operating in. Placing the tags can be done ahead of time if there is control over the environment or it can be done by another robot which does possess high quality localization. For the MR-SLC experiments we had a UGV running Simultaneous Localization and Mapping (SLAM), meaning it had high quality localization in the environment we were operating in. The way we used the UGV to assist the UAV was we mounted an AR bundle on top of the UGV and put a camera on the UAV. This meant as long as the UGV stayed within line of sight of the UAV the UAV could accurately deduce its pose relative to the one the UGV was broadcasting, thus allowing the UAV to perform any metric motion that kept it within range and sight of the UGV. Another option is to have a camera or another robot to look at the robot that needs localization assistance and tell it where in the environment it is. This option still requires the location of the assisting robot or camera's location to be known and cooperating with the uncertain robot. We did not use this option for MR-SLC because it required either putting tags on the UAV (adds weight) or writing an algorithm to identify the pose of the UAV (a lot more work and not the focus of this project). In addition to those problems it required making it so the camera on the UGV could always see the UAV, which would be difficult because the UAV flew at a higher height than the UGV.

3.2.5 Other Controllers

More controller-based motion primitives that could have been added to the MR-SLC project, but was not, is a visual servoing controller. This controller can work in two different ways. The first is to be able to identify landmarks in a mapped environment and use their locations to localize yourself and once localized perform metric motions based off this

localization. An example of this is a door finding algorithm indoors, in this case a robot would be able to identify doors in its camera field of view and use the camera to drive to the door. Once the robot has gotten to the door the robot could switch to wall following motion primitives or some other controller-based motion primitive. The other way to do this would involve having prior key frames in an environment, which would involve having taken a camera through parts of the environment and to have saving the images with lots of unique features and the locations at which these images were taken. Once this is acquired then the next time the robot comes through the environment, given the camera parameters, the robot will be able to match features with these key frames and use that to localize itself. The benefits of this over visual SLAM is that it is simpler and less computationally intensive, so a weaker computer is needed, which makes this viable for a large set of robots.

3.3 Experimental Analysis

For the analyzation of the MR-SLC system an additional planning capability was added to the SLC system. Planning with adaptive dimensionality (PAD) [35] allows for the planner to try and plan a path for the UAV first and then check to see if it needs to plan for the UGV in order to assist the UAV in its goal of navigating from one area to another. This system was augmented with the SLC framework in order to allow for planning with a variety of controllers. The planner was allowed to plan metric motions for the UAV if it maintained line of sight with the UGV and stayed within a certain distance of the UGV. The planner was also allowed to plan wall following motions and corner following motions. The only metric motion that the planner was allowed to call without localization was the yaw controller in order to turn and face the correct direction for the next controller in the plan. Table 1 below shows the planning time for PAD versus without and shows that it was needed for the experiments to be successful.

Table 1: MR-SLC Planning Time Results

Algorithm	Planning Time (s) Avg.	Std. Dev.	Num. Iter. Avg.	Num. Expansions Avg.	Path Cost Avg.	Final Eps. Avg.	Success Rate (%) @10min
PAD MR SLC	13.87	20.04	1.41	2405	13754	1.25	100
Full-D ARA* MR SLC	30.19	76.84	n/a	5388	14069	1.35	67.5

The planner operated by selecting which controller was next in the plan and then calling that controller with the specified end trigger. An example plan is shown below in Figure 5. In this example it can be seen which motion primitives belong to which controller and the area where the planner had to plan for the UGV in addition to the UAV is shown as a series of metric motions.

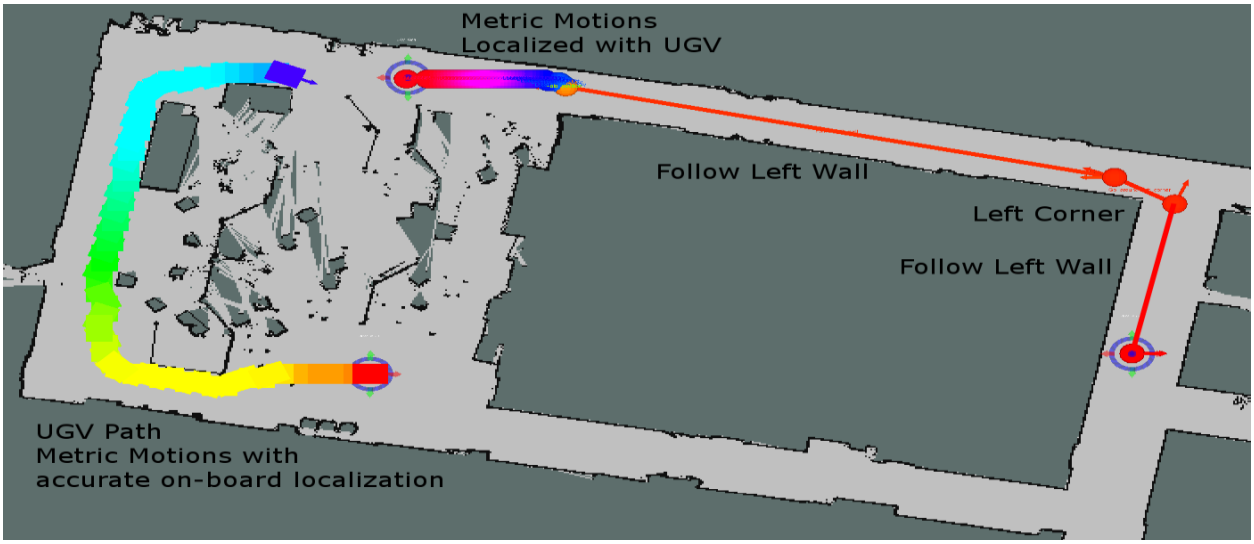

Figure 5: Example plan in the MR-SLC Experiments

In the above scenario, the UGV cannot provide localization assistance in the hallways on the right side of the figure, which forces the planner to try and find another way to get the UAV from its starting location (on the right) to its goal location (on the left). This plan is made up of a series of controllers starting with follow left wall until opening on the left, follow left corner, follow left wall until opening on the left, and finally metric motion to the goal. The metric motions require the UGV to move from the bottom left to the top left in order to be close to the UAV and be within the line of sight of the UAV. While testing the system the motion controllers all worked robustly as intended and the planner generated solutions, which utilized the SLC framework. Overall this experiment was a success because the system worked and was accepted to ICRA 2016, where it was presented.

4. Height Estimation for UAV in Air-Ground Systems

For UAVs it is incredibly important to have a very accurate measurement of the pose of the robot because crashing can be very catastrophic for the system. In indoor environments this problem is even more crucial, especially the measurement of the height because if the UAV hits the ceiling it could crash. Height estimation is normally done by having a point range finder facing the ground and combining that data into a filter with other sensors, such as an IMU, and a barometer. Both the IMU and barometer are noisy in indoor environments so much of the height estimation falls to the range sensor. In flat ground environments this is a decent, cheap solution to the problem of how high above the ground is the UAV. However, indoors environments are rarely flat due to objects being in the environments, such as chairs, tables, and other loose objects. In these environments it is possible to try and use the IMU and range sensor to try and detect these objects and represent them as offsets from the ground. This solution works best with flat objects that are a significant distance off of the ground, but this solution is still prone to errors and a single bad height offset estimation can lead to a failure of the whole system.

Another solution is to use visual odometry to try and measure the height of an agent. This is a very good solution when given a feature rich environment and can be combined with other sensors, like listed in the previous paragraph to give a fairly accurate measurement of the height and pose of a robot. This method depends on the environment containing useful features and this may not always be the case. There could easily be an environment that has object on the ground and a lack of unique features making visual odometry hard to use. In this case we would like to be able to provide an accurate height estimation regardless of the environment's topology and even be able to hand even more complex topologies such as ramps and staircases.

4.1 Probabilistic Elevation Map

Given the above height estimation problem and a large highly capable UAV we have design a system that takes advantage of the sensors available to our UAV in order to build an elevation map which can then be used to accurately estimate the height of the UAV anywhere in the environment. This method requires an UAV with a LIDAR that scans a vertical slice of the world, including the ground and uses that information to build a probabilistic elevation map (PEM). The elevation map is created and updated and added to before the UAV launches and will continue to update and build upon itself throughout the entire flight.

As mentioned in the related work, this Probabilistic Elevation Map works in a similar way to that of occupancy grids and could almost be considered an extension of occupancy grids. The way that occupancy grids work is that they try and make a 2D occupancy map of the environment. These grids are made up of discretized cells and have a binary value associated with them as to whether they have an obstacle in them or not. The Probabilistic Elevation Map works very similarly but instead of a binary value denoting the presence or absence of an obstacle the PEM has a real valued number that represents the height of a given cell.

Before the elevation map can be built the transform between the map and the body frame of the UAV needs to be established. This can be done in a few ways, the first being the easiest and least robust. This requires knowing the height the UAV starts at and hardcoding it into the transform at the beginning. This method is poor because it is robot specific and could fail very badly if the robot ever started at a much different height. The second method is to use a vertical LIDAR or a point range finder to read how far away the ground below the UAV is from the sensor and use that to initialize the height. This approach is good as long as the UAV always takes off from the ground or close to the ground and this approach could fail if the UAV is launched from a platform or surface that is lifted off the ground plane. The way we get the initial height is the most robust way, is to take a vertical scan and try and find the plane that best fits what is expected from the ground plane. This means to try and find the LIDAR scans that have the lowest height in body frame coordinates and to try and fit a line to those points such that the line has relatively little deviance in height but changes any amount in x and y. Once this line is found it is trivial to use the scan and ground plane to find the approximate height of the UAV. The only ways in which this method could fail is if the LIDAR cannot get any (or too few) points that are actually on the ground or if there are holes in the actual ground and these holes might get mistaken for the actual ground. Both of these failure methods should be considered before launching the UAV because a bad height estimation could cause a great deal of problems during any missions.

Once this initial transform is established it becomes possible to safely initialize the elevation map and start adding scans to it. This is done in a few steps; the first step is to initialize the elevation map. The second step is to filter and transform the latest LIDAR scan to the map frame. The third step is to take the transformed scan and match it with the current elevation map. The fourth step is to update the elevation map by adding in new points and updating recently seen points. The last step is to decay the map and publish the new estimated height of the UAV. Initialization is done once the initial height is established by the most recent scan. This scan is transformed into the map frame and discretize into a rectangular grid that is sized to fit all of the points in the scan in it. Once the grid is initialized the scan is added in. Each scan point is fit into a cell in the grid; cells have three variables they track: the height of the cell, the variance of the height of the cell, and whether the cell has been seen by a scan or not. The variance of each cell in the initial map is proportional to the distance it is away from the LIDAR meaning closer points have a lower variance and farther points have a higher variance and their height reading is less certain.

After initialization is over the elevation map is used in the height estimation routine, which is called every time there is a new scan. The update starts by filtering the scan to only return scan points that are at least 30cm away from the UAV. This number is an UAV specific parameter which keeps the UAV from using points of the scan that hit the body of the UAV. The update also filters points that are on the ceiling by filtering out points that are more than 90 degrees off the normal of the ground plane. Once filtering is complete the scan is transformed to the map frame and the distance between each point and the LIDAR is saved alongside the 3D point in map frame.

Once the points and ranges are saved the points are then matched to the current elevation map. This can have two different meanings depending on the number of points that are within a threshold of the points in the PEM. If a significant number match, then the weighted average offset between the PEM and the scan is found and is used to find the change in height from the last scan. If few points match, then that means the scan is hitting a mostly unseen area at which point a similar method to getting the initial height is used and the height estimate is filtered with the previous height estimate. The change is then saved and used in the next step in updating the PEM.

The PEM is updated in two steps: changing the PEM's size, and then adding in the new scan to the map. Updating the map's size is trivial because it just saves the current data and then the size is updated to include minimum and maximum off all the points. Adding in new points comes in two forms: unseen cells, and seen cells. If the cell is previously unseen then it is set to seen, the point's height (using the new found UAV's height as the baseline) is saved as the cell height, and the range to the cell from the UAV is used to get the variance of the cell. If the cell has been seen, then the point's height is compared to the cell's height and combined if within a certain threshold. This combination is based off the cell's variance and height and the point's variance and height. If the heights are significantly different then that could indicate that a dynamic object could have moved into the cell. Since this uncertainty is introduced of whether there is now an obstacle or not in a certain cell the cell's variance is then vastly increased and when the variance gets large enough the cell's height is updated to be the newly scanned height, whether that is the addition of an object in a cell or the removal of an object from that cell.

After the map has been updated by the latest scan every cell in the map then is decayed. This decaying takes the variance of every seen cell and makes it slightly larger. This is used to show that areas that have not been seen for a while have more uncertainty than cells that are regularly updated. After the map is decayed then the latest height estimate is published and the system waits for the next LIDAR scan to repeat the above steps (other than initialization).

4.2 Experimental Analysis

The probabilistic elevation map was tested in simulation against a naïve median filtering of the LIDAR points. This median filter was used on the 10 points with the lowest elevation in the map frame when transformed from the sensor frame. The PEM can be seen pre-takeoff in Figure 6 below.

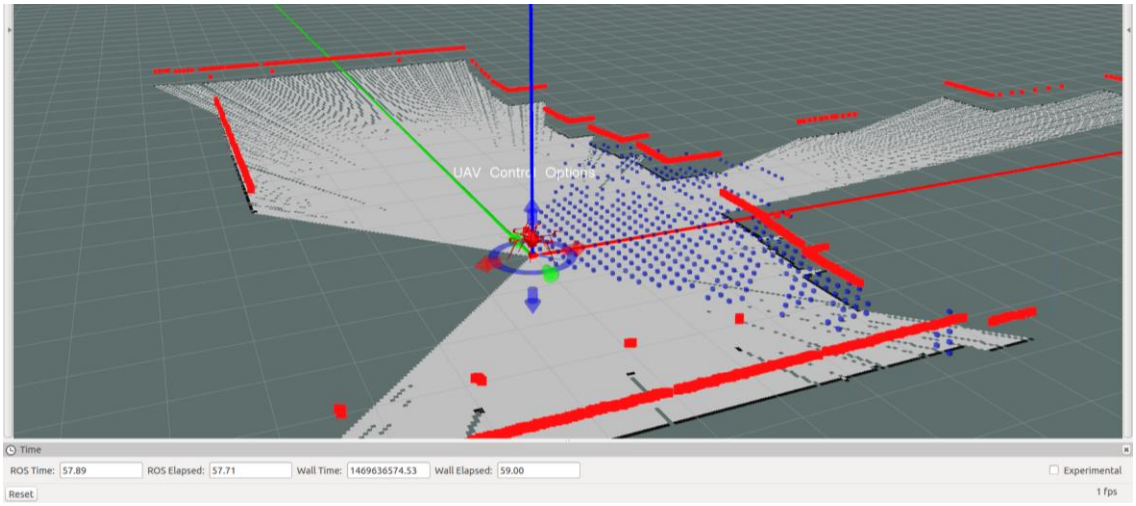

Figure 6: PEM Pre-takeoff

In the figure above the UAV is shown in red on the ground at the tip of the triangular arrangement of boxes. The boxes represent seen cells in the PEM and they are shown at their estimated height in the map. It is important to note that the seen cells are in a triangular arrangement because the LIDAR is servoing over a 70-degree arc. The map simulated environment which was used for the PEM verification and tests is shown below in Figure 7. In this environment the UAV starts on the ground and is in an environment with boxes on the ground. There are also bleachers and barriers present in the world.

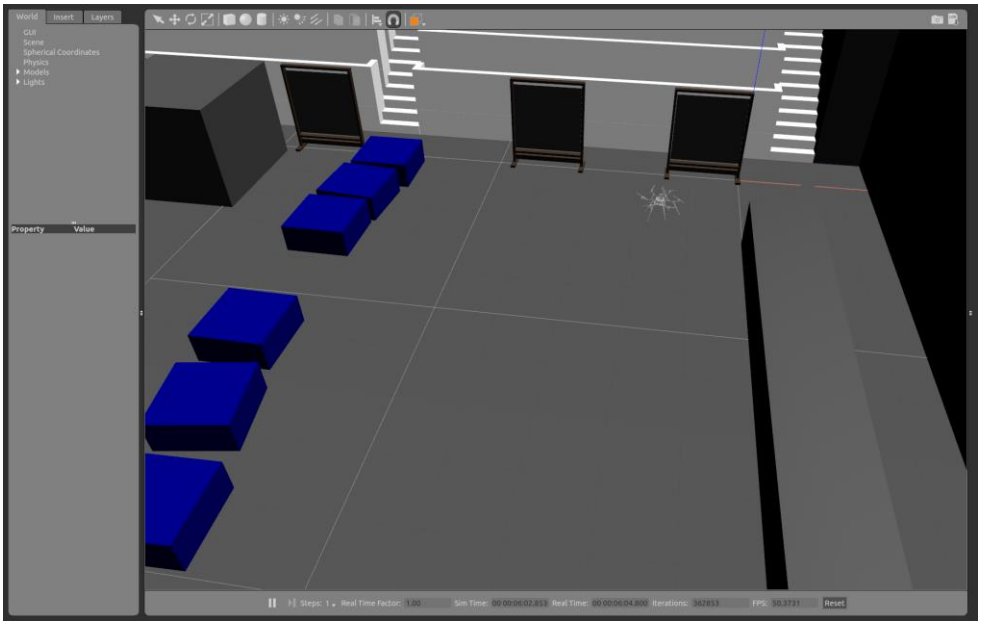

Figure 7: Simulated Test Environment for the PEM

The PEM was successful at initializing the map and all of the data was able to be filtered correctly which ultimately led to the flight test, which was far more stable than the naïve method. The error and standard deviation from ground truth are shown when flying over flat ground in

Table 2 below, the data is shown again when flying over boxes in Table 3 below, and once more the error data is shown when flying over a large ramp in Table 4. The data shown was taken in simulation a represents the data before it was put in a low pass filter with the previous reading's estimate.

Table 2: PEM versus Naïve Error When Compared to Ground Truth Flying Over Flat Ground

Algorithm	Average Error	Std Dev Error	Minimum Error	Maximum Error
PEM	0.0028 m	0.0076 m	-0.0545 m	0.0471 m
Naïve Median	0.0925 m	0.1429 m	-0.0484 m	1.2385 m

Table 3: PEM versus Naïve Error When Compared to Ground Truth Flying Over Boxes

Algorithm	Average Error	Std Dev Error	Minimum Error	Maximum Error
PEM	0.0025 m	0.0075 m	-0.0481 m	0.0737 m
Naïve Median	0.0654 m	0.1162 m	-0.0440 m	0.9126 m

Table 4: PEM versus Naïve Error When Compared to Ground Truth Flying Over a Ramp

Algorithm	Average Error	Std Dev Error	Minimum Error	Maximum Error
PEM	-0.0062 m	0.0285 m	-0.1006 m	0.0776 m
Naïve Median	0.0526 m	0.2232 m	-0.8322 m	1.5114 m

The hover elevation map is shown below in Figure 8. In this figure it is easy to pick out the flat surfaces that make up the ground and the tops of the boxes in the environment.

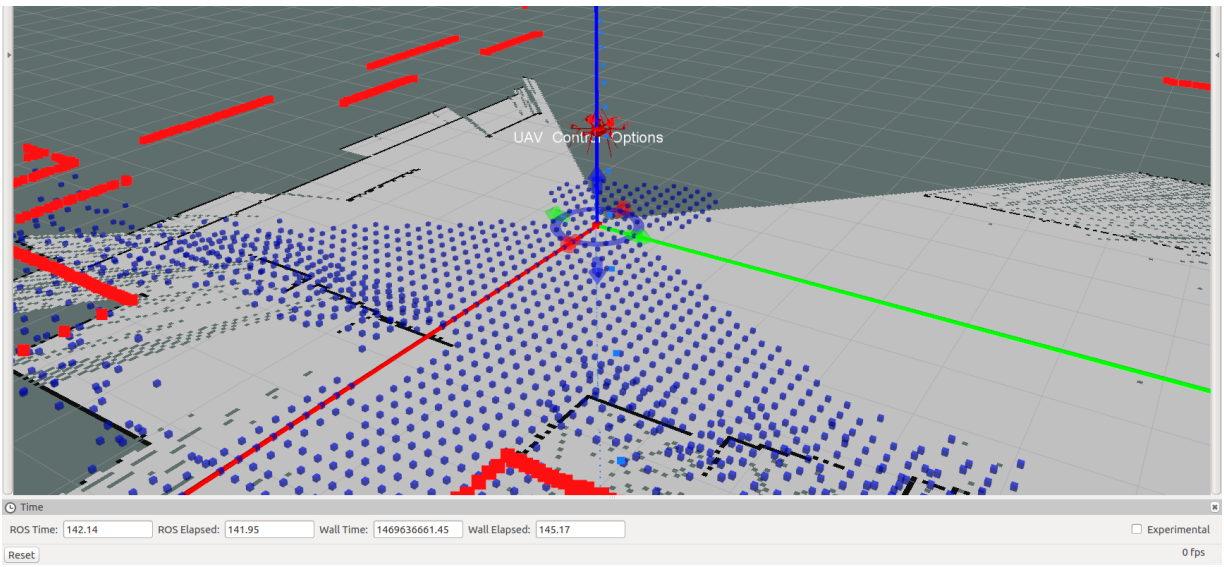

Figure 8: PEM While Hovering

The height fluctuations of the height estimation were significantly improved when using the PEM as opposed to the naïve approach. This held true for hovering, flying over flat ground,

flying over boxes, and moving boxes underneath the hovering and moving UAV. The PEM was a useful improvement over the naïve approach and can be extended to many more applications involving active state estimation of UAVS.

5. Summary of Contributions

In this work it is shown the contributions made to motion control, in the form of viable, lightweight controllers which can generate controller-based motion primitives in arbitrarily complex environments with arbitrary triggers to end them. These implemented controllers include a metric motion controller, a wall following controller, a corner following controller, and an AR assisted metric motion controller. The metric motion controller generates viable metric motions for a UAV based off desired motion, when localization is available. The wall following controller makes it simple for any robot with the correct sensing capabilities to follow a wall until an opening on either side or the wall reaches an ending in the form of an inside corner. The corner following controller generates controller-based motion primitives to navigate an UAV around corners in hallways or rooms. Lastly the AR assisted metric motion controller allows for UAVs to get assistance in localization from an outside source, which then makes metric motions viable in these situations. All of these controllers were crucial to the success of our paper [34], which I presented at ICRA 2016.

In addition to our work on MR-SLC, SBPL's Hexacopter (UAV) and Segbot (UGV) exploration team was in need of a robust height estimation system for the UAV, which would work in dynamic environments (the UGV drives underneath) and allow for the UAV to take off from the UGV while obtaining and maintaining a height estimation with minimal error. I completed this task and created a probabilistic elevation map (PEM) system, which allows for a probabilistic height estimation of the UAV throughout a map of seen cells, which is constantly being updated in an efficient manner. This height estimation allows for the UAV to have a better knowledge of what it is seeing and what it expects to see beneath it as it travels around and completes its primary objectives, such as exploring a new environment.

6. Conclusion

Two main projects have been shown here and how they relate to autonomous flight and navigation in different air-ground systems. In the first work it was shown that simplistic UAVs with minimal capabilities can uses the SLC framework in order to navigate an environment with the use of different controller-based motion primitives. This work served as the basis for our MR-SLC experiments, which paired a UGV with excellent localization capabilities with one of these UAVs that cannot localize without help and showed that it was possible to plan for both of these robots to navigate an environment through the use of a prior map. This work should further be explored with the addition of new controllers such as a visual servoing controller or a keyframe-based visual odometry controller. Another extension of this work would be to try and add additional robots into the environment and see what the capabilities of the new system would be. Furthermore, it would be useful to try and extend this system to an environment without a prior known map and use these controller-based motion primitives along with help from other heterogeneous robots to explore and search for an object of interest.

The second work introduced in this paper was the addition of a probabilistic elevation map, which is used by a UAV in accurately determining the height of the robot as it travels through the environment. This work is a practical approach to efficiently using a LIDAR in height estimation and provides a simple algorithm with which the system could deal with dynamic obstacles and maintain accurate probabilistic height estimations. This work can be extended into the navigation field and used in UAVs which don't need to plan for yaw while in transit (fixed wing UAVs need to account for yaw, but multicopters generally do not). These UAVs can use the PEM to change their heading as they fly such that they obtain the best LIDAR scans possible to maximize useful data in estimating the height of the UAV. Another extension would be to add a smart takeoff and landing system to the UAV. This means that the UAV would use the PEM to make sure that when it was taking off (from atop a moving surface for example) or landing that it was not too close to any objects so as to minimize the aerodynamic effects of these objects. This would allow for smoother, safer operations with little extra computational overhead. The functionality of the PEM can also be extended to have a base plane instead of a ground plane and this could allow for mapping areas that have a lower elevation than the ground. Also recent works in elevation maps have added in the functionality to track multiple layers [31], such as the top of a table and beneath the table, which would be useful because the LIDAR scan could see both areas and get conflicting readings. In summary, both of these works, MR-SLC and PEM, provide useful additions to existing works and ideas and can be used to further autonomous flight and navigation in air-ground systems.

References

[1] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister and M. Likhachev, "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 258-265, 2014.

[2] V. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE Transactions on Automatic Control,* vol. 31, no. 11, pp. 1058-1063, 1986.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE International Conference on Robotics and Automation*, 1985.

[4] C. I. Connolly, J. B. Burns and R. Weiss, "Path planning using laplace's equation," in *IEEE International Conference on Robotics and Automation*, 1990.

[5] J. S. Lewis and J. M. O'Kane, "Planning for provably reliable navigation using an unreliable, nearly sensorless robot," *International Journal of Robotics Research,* vol. 32, no. 11, pp. 1339-1354, 2013.

[6] A. A. Masoud, "Managing the dynamics of a harmonic potential field-guided robot in a cluttered environment," *IEEE Transactions on Industrial Electronics,* vol. 56, no. 2, pp. 488-496, 2009.

[7] R. C. Arkin, "Motor schema based mobile robot navigation," *The International Journal of Robotics Research,* vol. 8, no. 4, pp. 92-112, 1989.

[8] R. R. Burridge, A. A. Rizzi and D. E. Koditschek, "Sequential conposition of dynamically dexterous robot behaviors," *IJRR,* vol. 18, no. 6, pp. 534-555, 1999.

[9] D. C. Conner, H. Choset and A. A. Rizzi, "Integrating planning and control for single-bodied wheeled mobile robots," *Autonomous Robots,* vol. 30, no. 3, pp. 243-264, 2011.

[10] V. Kallem, A. Komoroski and V. Kumar, "Sequential conposition for navigating a nonholonomic cart in the presence of obstacles," *IEEE Transactions on Robotics,* vol. 27, no. 6, pp. 1152-1159, 2011.

[11] H. Kress-Gazit, G. E. Fainekos and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *IEEE International Conference on Robotics and Automation*, 2007.

[12] R. C. Arkin and T. Balch, "Aura: principles and practice in review," *Journal of Experimental & Theoretical Artificial Intelligence,* vol. 9, no. 2-3, pp. 175-189, 1997.

[13] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trasactions of the ASME Journal of Basic Engineering,* vol. 82, no. Series D, pp. 35-45, 1960.

[14] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," Australian Centre for Field Robotics, Sydney, Australia, 2006.

[15] A. Bry, A. Bachrach and N. Roy, "State estimation for aggressive flight in GPS-denied environments using onboard sensing," *IEEE International Conference on Robotics and Automation,* pp. 1-8, 2012.

[16] I. Dryanovski, W. Morris and J. Xiao, "An open-source pose estimation system for micro-air vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, 2011.

[17] P. Nunez, R. Vazquez-Martin, J. C. del Toro and A. Bandera, "Feature extraction from laser scan data based on curvature estimation for mobile robots," in *IEEE International Conference on Robotics and Automation* , Orlando, FL, USA, 2006.

[18] X. Pan, D. Ma and Z. Jiang, "Vision-based approach angle and height estimation for uav landing," *Congress on Image and Signal Processing (CISP),* vol. 3, pp. 801-805, 2008.

[19] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," *International Symposium on Robotics Research (ISRR),* vol. 2, 2011.

[20] N. Engelhard, F. Endres, J. Hess, J. Sturm and W. Burgard, "Real-time 3d visual slam with a hand-held rgb-d camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden, 2011.

[21] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," *IEEE International Conference on Robotics and Automation (ICRA),* pp. 4531-4537, 2011.

[22] S. Weiss and et al, "Monocular Vision for Long-term Micro Aerial Vehical State Estimation: A Compendium," *Journal of Field Robotics,* no. 5, pp. 803-831, 2013.

[23] N. Trawny and et al, "Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks," *Journal of Field Robotics,* vol. 24, no. 5, pp. 357-378, 2007.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," *IEEE and AMC International Symposium on Mixed and Augmented Reality,* pp. 225-234, 2007.

[25] R. A. Newcombe and et al, "Kinect Fusion: Real-time dense surface mapping and tracking," *IEEE Symposium on Mixed and Augmented Reality (ISMAR),* 2011.

[26] R. Mur-Artal, M. M. J. Montiel and J. D. Tardos, "Orb-SLAM: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics,* vol. 31, no. 5, pp. 1147-1163, 2015.

[27] J. Engel, T. Schops and D. Cremers, "LSD-SLAM: Large scale direct monocular SLAM," *European Conference on Computer Vision,* pp. 834-849, 2014.

[28] Y. Cang and J. Borenstein, "A new terrain mapping method for mobile robots obstacle negotiation," *AeroSense,* pp. 52-62, 2003.

[29] P. Pfaff, R. Triebel and W. Burgard, "An efficient extension to elevation maps for outdoor terrain mapping and loop closure," *The International Journal of Robotics Research,* vol. 26, no. 2, pp. 217-230, 2007.

[30] B. Steder, G. Grisetti, C. Stachniss and W. Burgard, "Visual slam for flying vehicles," *IEEE Transactions on Robotics,* vol. 24, no. 5, pp. 1088-1093, 2008.

[31] R. Triebel, P. Pfaff and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 2276-2282, 2006.

[32] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots,* vol. 15, no. 2, pp. 111-127, 2003.

[33] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine,* vol. 9, no. 2, pp. 61-74, 1988.

[34] J. Butzke, K. Gochev, B. Holden, E. Jung and M. Likhachev, "Planning for a Ground-Air Robotic System with Collaborative Localization," *IEEE International Conference on Intelligent Robots and Systems (IROS),* 2016.

[35] K. Gochev, B. Cohen, J. Butzke, A. Safonova and M. Likhachev, "Path planning with adaptive dimensionality," in *Symposium on Combinatorial Search (SoCS)*, Barcelona, Spain, 2011.