Final Year Project Report

Full Unit - Interim Report

Autonomous Micro Air Vehicles: Enhanced Navigation in GPS denied environments.

Roger Milroy

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science (Artificial Intelligence)

Supervisor: Professor Sara Bernadini

Department of Computer Science
Royal Holloway, University of London

February 25, 2020

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 6028

Student Name: Roger Milroy

Date of Submission: December 6th 2019

Signature: Roger Milroy

Table of Contents

Abstract

All robotic systems that are mobile require an understanding of their position in order to maneuver effectively, avoid obstacles and carry out useful tasks. In the context of Micro Air Vehicles (MAVs) the situation is more challenging due to the twin problems of noisy sensors and limited computation. There are many different approches to solving this problem and most if not all of them use the Kalman Filter in order to fuse sensor data and provide optimal estimates of state.

The main objective of this project is to implement the newly proposed technique of Hybrid Inference (HI) as an improvement on the Kalman Filter in the context of position estimation in MAVs in a simulated environment. The secondary aim is to demonstrate it on the DJI Matrice 100 with the Jetson Nano as the onboard computation platform.

The project is based upon a technique of Monocular Simultaneous Localization and Mapping (SLAM) which also recovers absolute scale. This integrates visual data and absolute measurments, such as Inertial Measurment Unit (IMU) data. It uses an Extended Kalman Filter to integrate the sensor data which is what this project will be replacing and evaluating the impact. My initial assessment of the Monocular SLAM technique was that the technique was useful but implemented on a specific platform, the Parrot AR drone. Upon further investigation I discovered that the implementation was mostly platform agnostic. This led me to prioritise HI over reimplementing the Monocular SLAM technique.

The first terms work has not provided much opportunity for applying Software Engineering principles due to the bulk of the work being that of mastering new theory, learning new technologies and assembling existing projects. I was however able to apply them to the design of the architecture of the system as a whole and in the HI demonstrator.

Monocular SLAM with scale recovery is the technique upon which hybrid inference will be demonstrated. Kalman filters are the existing solution and also form the basis of the hybrid inference solution, as it uses the same problem formulation. HI is the general technique that fuses existing mathematical techniques with neural networks. The report explores both theoretical and technical aspects of all these techniques.

The report details a summary of the work that has been completed this term. Learning how to use ROS and Gazebo, setting up the TUM ardrone and hector packages that I am building upon. And finally, implementing a demonstrator of the HI technique on a synthetic dataset.

Some additional challenges have been identified during the course of this work. First is that HI as described in the paper, deals with a simplified linear dynamic model that does not have inputs to the system. The dynamic model that describes MAVs must include external inputs for control. Reformulating the graphical model to account for this is the first new challenge that I had not identifiend in the initial project plan. The second challenge is temporal. With a Kalman Filter, it is possible to estimate future states and the uncertainty of those postulated states. This is critical for dealing with latency issues in the real time control environment that MAVs present. The current formulation of the underlying graphical model does not allow for future reasoning, solving this is necessary to enable HI to be used this domain.

Finally I describe the work to be completed in the second term. The provided Gantt chart shows the projected timeline.

Chapter 1: **Introduction**

1.1 Motivation

Micro Air Vehicles (MAVs) popularly known as quadcopters, have become ubiquitous in recent years with prices dropping across a range of sizes of drones. This has lead to their deployment across a number of sectors and applications. These include videography, where most of us will have seen their output, all the way to assessing and counting endangered species with numerous other applications in between.

One major challenge is that of fixing position accurately. Most solutions use some kind of satellite navigation solution to fix absolute position. This is effective in outdoor environments but ineffective in a number of environments, such as anywhere indoors, in forests or underwater. In these environments it is necessary to rely on inertial sensors as well as image or other data. These are fused in an Extended Kalman Filter to produce a single estimate of position. The main issue with these techniques is that absent regular fixes of known positions, estimates of position accumulate error.

If we can reduce the rate that this error accumulates, this opens up the space of viable environments that automated systems can operate in and increases their usefulness.

1.2 Objectives and Contributions

1.2.1 Objectives

The main objective is to apply the newly proposed technique of Hybrid Inference (HI) [26] to the problem of position estimation in MAVs. HI integrates existing mathematical models with Neural Networks in order to improve inference performance in both low and high data regimes.

In addition I aim to demonstrate HI using onboard computation, taking position estimates that come from inertial sensors and visual SLAM systems. I will use a Jetson Nano for computation, mounted to a DJI Matrice 100 quadcopter.

For the first term, the key objectives were to validate the technique by implementing it on a small synthetic dataset, to demonstrate in Gazebo the Monocular SLAM with scale recovery technique [11] upon which the Hybrid Inference will be implemented and finally to validate the Jetson Nano platform for computation. All these objectives were successfully achieved.

The objectives for the second term are to implement Hybrid Inference for state estimation completely and demonstrate it on a simulated quadcopter in Gazebo. Secondary goals are to integrate the Jetson Nano onto the Matrice 100 and adapt the code to work with the DJI OSDK. Finally to demostrate the fully realised project in precision figure flying in order to compare with the results of Engel et al. [10].

Chapter 2: **Professional Issues**

2.1 Professional Issues

The core ethical concern for me during this project is that of application. The application of technology is where the bulk of the societal impacts occur. But for the people developing these technologies there are two somewhat distinct situations. There is the case where we develop a particular application, like Facebook which has a particular direct impact on the world which we can then evaluate as to whether it is beneficial or not. Then there is the case where we develop a technology that underlies applications and maybe enables new features or improves the efficacy of existing features, we could also think of these as tools used for creating other applications. This situation is harder to evaluate as to whether the technology is beneficial or not. A good example of this is the Internet, as a technology it is neutral but it has enabled many applications that people would see as mostly good such as improving access to learning for hundreds of millions of people. At the same time it has enabled criminals to perpetrate new forms of crime such as blackmailing people over intimate photos they have gained access to illegally. It is not clear how to weigh the relative merits of the applications so it is also hard to evaluate the technology itself. One of the core challenges of this situation is whether there is a responsibility to mitigate the risk of negative uses of the technology.

I would consider my project to fall into the second category. This is because the primary goal of this project is to demonstrate a technique that can reduce errors in pose estimation. This could enable more precise positioning of drones for longer periods of time, which could open up some environments that currently are more challenging for drones to operate in. The main examples are indoors, in forests or anywhere where GPS coverage is intermittent, including cities with a large number of skyscrapers in close proximity, and finally underwater. Thus it is an enabling technology rather than an application of existing technology.

– Positive applications REDO... I want to explore some of the potential applications of my project, both what I would consider to be beneficial and those I would consider undesirable. On the positive end of the spectrum are applications like automated search and rescue. By reducing dependence on GPS signals, in the best case drones could be deployed fully autonomously to locate missing persons in a variety of domains where they cannot currently and if done with large numbers of drones it could drastically reduce the time taken to find missing persons. Another positive application could be in increasing reliability and accuracy for consumer navigation applications. This could improve the customer experience by reducing the amount of erroneous instructions given by these applications. It is also possible that by improving localisation accuracy indoors that a wider range of indoor robotics applications could be enabled. This is towards the bottom because there are a wide array of other challenges that must be overcome and navigation accuracy is not the greatest of them.

– Negative applications There are in contrast a number of potentially harmful applications. These are mostly concerned with the enabling of more autonomous weapons. Currently there are many consumer drones deployed with some measure of autonomous capability such as Return-to-Home capabilities. We have not yet seen any military use of autonomous MAVs however improved precision of flight may open up this area. A small thought experiment we could carry out would be to think about the incident at Gatwick in December 2019 where around 1000 flights were disrupted [4] and considering how that would have been different if the drones involved were autonomous. And again if there were multiple, it would be extremely difficult for the authorities to effectively prevent interference with airport operations. To extend the thought experiment, imagine that the drones were fitted with explosives. This is

not too much of a stretch given other events such as those in Venezuela in August 2018 [3] where that was exactly the assertion.

Looking at military capabilities and plans, it is very likely that swarming drones will be put into service in the not too distant future. This is due to two primary factors. One is cost. The unit cost of military hardware in the UK increases at a high rate with unit costs for RAF aircraft increasing at 11.5% [18]. This means that the numbers of vehicles, ships and aircraft has been falling. Drones have the potential to reverse this trend by equipping large expensive human operated vehicles, ships and aircraft with large numbers of cheaper, smaller drones. There has been much talk of swarming drones to overwhelm conventional defences and this is one expression of the interest the military has in autonomous drones.

At the same time the technologies enabling the autonomous operation of drones also lowers the barriers that prevent smaller but malicious actors from using them to cause harm. In fact ISIS and other terrorist organisations have already been known to use remotely operated drones to carry out attacks [**?**]. It is not that big of a step for them to deploy them autonomously. In many cases the tools are there to program a predefined flight path.

What is not so clear is to what extent improving navigation accuracy enables autonomous operation. It is a core competence of an autonomous device [**?**] but not the only one. And depending on the environment and desired accuracy it can be considered a solved problem. If operating outdoors and needing accuracy of around 1m, GPS navigation is sufficient. It is only in more complex terrain and with higher accuracy requirements that you may struggle. In my opinion there are other more important enabling factors, the most important of which are the tools that support the development of autonomous behaviour and simplify that process.

In conclusion, I think that there are risks of negative outcomes given this technology however I don't think that it enables any greater risks than already exist and at the same time I don't think it increases the likelihood of any other risks appreciably.

A lot depends on the efficacy, and efficiency of the solution. In all likelihood it is unlikely to be much more effective than the EKF and is more computationally expensive than is feasible for small scale applications. This limits the risk considerably.

Chapter 3: **First Term Planning**

3.1 Original Plan

The original plan delayed the bulk of the implementation until the second term. While this was not unreasonable given the learning curve for each of the technologies involved, if I stuck to the original plan I would have been faced with a huge amount of work with little to base it on. There was also quite a lot of uncertainty about the state of the TUM codebase. The original assumption was that the code had been written specifically to work with the Parrot AR drone as it was the platform on which they demonstrated [11]. This would have meant rewriting the code to work in ROS and be platform independent. When I found that was not the case it changed the dynamics of the project quite significantly and led to the new plan.

I have included below the original plan Gantt Charts that detail milestones and timescales.

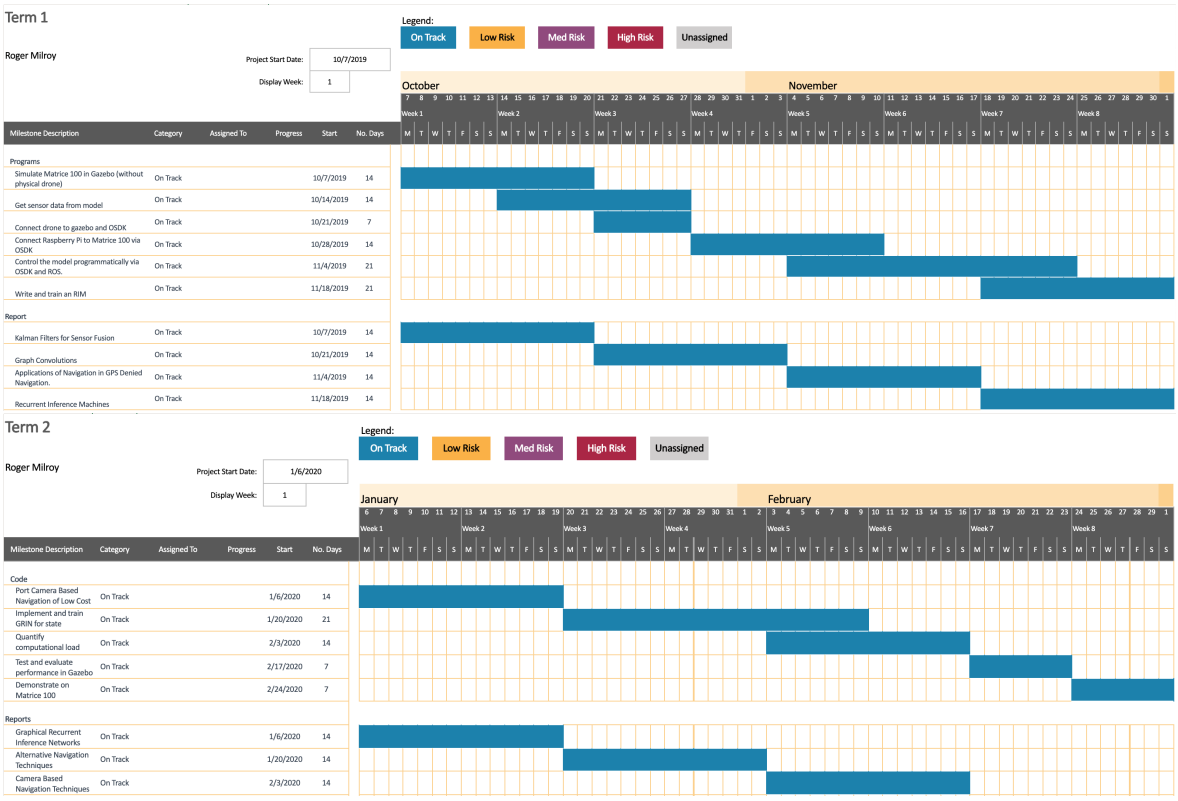

Figure 3.1: Original Plan Gantt Charts

3.2 Revised Plan

While working on the project the outlook changed quite significantly when I found the code for the Monocular SLAM with scale recovery was open source. This meant that I could directly build upon their work and extend it instead of spending the bulk of the project reimplementing their technique. This enabled me to pivot to focusing more on the implementation of HI.

Under the new plan most of the first term is spent getting simulation working with the existing code base. I also implement a simple version of the HI technique on synthetic position data.

I replanned and the following are the updated milestones and associated dates. I will elaborate on the work to be done in the second term at the end of the report.

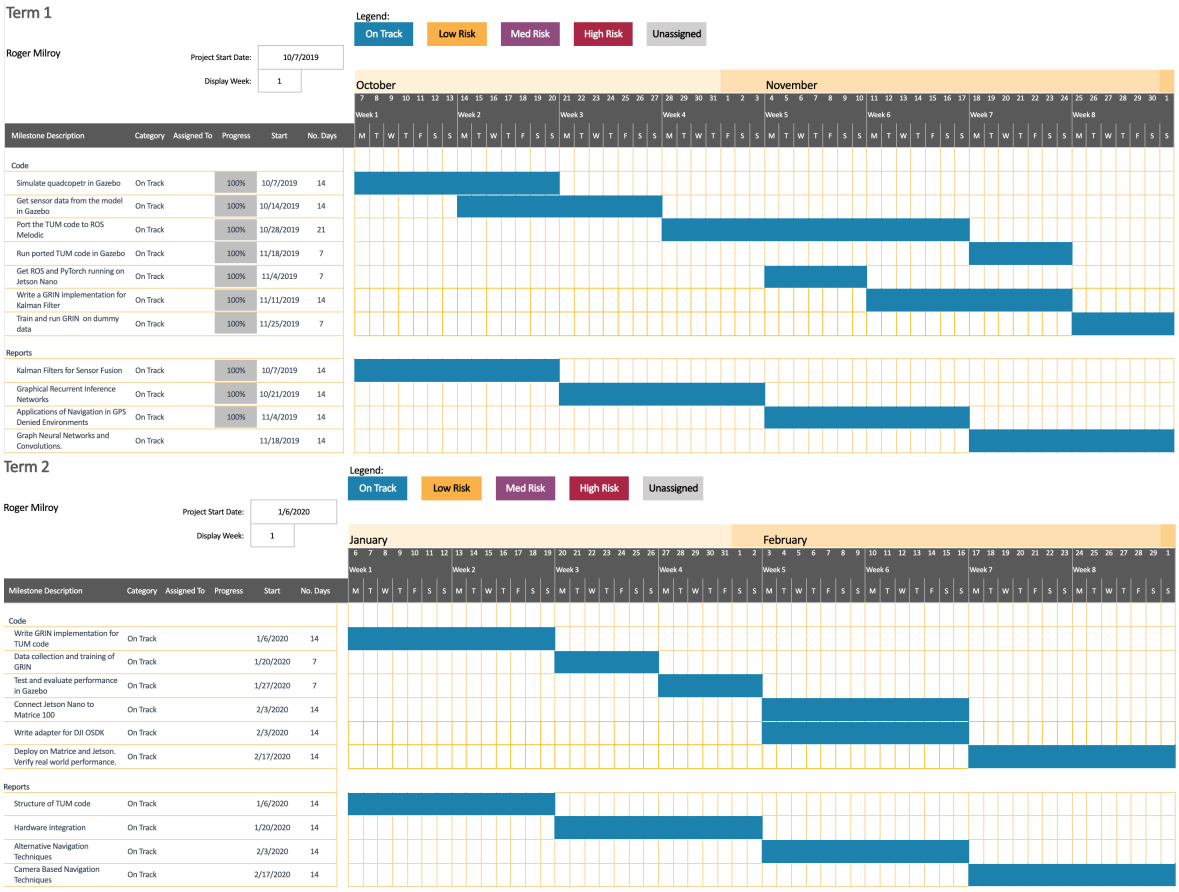

Figure 3.2: Revised Plan Gantt Chart

Chapter 4: **Software Engineering**

4.1 Tools

4.1.1 Version Control System

I used git as my version control system, and GitHub as my remote. I actually have a number of repositories because for the ROS section of the project it was necessary to modify existing projects code. I created a private fork of the relevant projects and then modified them as necessary.

I have separate branches for development, reports and for each feature that I am working on. These last ones are only temporary and are closed as soon as the feature is tested and integrated back into development. The workflow that I have decided upon is to use development for completed features once they have been tested. This does not imply that development is stable however so I only release code considered stable to master. The idea is that master should only contain stable code and be safe to use at any point.

4.1.2 Project Task Tracking

I used Trello in order to organise and keep track of tasks while completing them. I use a single board with To Do, Doing and Done lists. Each task is a card and has an associated due date. This is really useful to stay on track and quickly assess the state of the project at a glance. I also used this while replanning as I could evaluate each task and see whether it was still relevant in the new context. One additional advantage of using Trello is that it keeps track of history and has space for lists within tasks allowing them to be broken up into sub tasks.

4.1.3 Development Tools

I used both VSCode and PyCharm as Development Environments. They both have different advantages and disadvantages and I used VSCode for the C++ work and PyCharm for the Python work. For ROS development I have an Ubuntu VM with the relevant dependencies installed as well as the two IDEs I just mentioned.

I used the built in testing framework for Python, 'unittest' for testing. This provides a very similar framework to JUnit and allows for clear easy test setup and management.

4.2 Development

4.2.1 Design

ROS lends itself to modular code however the packages that I am working with and building on have very mixed engineering approaches. To date my work has been composing the

projects and getting them to run so Software Engineering practices have been limited to thinking about the design of the full implementation.

Unfortunately ROS is not particularly conducive to Test Driven Development (TDD) as there is no framework that enables testing outside of nodes. I can test the components of each node in the standard fashion, which I plan to do though this will apply more next term.

All of the design decisions that I have taken, both for implementations this term and the plans for future work, have been in the pursuit of modularity.

It is widely accepted that good code should be modular and reusable where possible. There are aspects where this is obviously not possible, such as configurations and any implementation specific work, but this should where possible be isolated and offer abstract interfaces.

With respect to the packages I have chosen, the hector stack and the TUM work, they have to some extent taken this approach but only really within the project. This complicates my work somewhat but at the same time using these as the base allows me to explore more advanced concepts over reimplementing the functionality of these modules. I will be writing adapters to enable the existing codebase to interact.

For the proof of concept of Hybrid Inference, the Dataset and the Linear Motion Model that creates synthetic data I implemented them as separate classes. A standard Object Oriented approach and used composition where necessary.

4.2.2 Testing Strategy

Due to the limitations that I have regarding support for testing frameworks, particularly in ROS, I have relied a lot on sanity checking at each stage while testing formally as many sections as possible. Integration tests are particularly important for my project as I have a variety of different technologies that need to work together across a number of interfaces. This will become important in the next stage of the project as that is when the different components start to interact and when I will be looking to start deploying code onto the hardware.

To this point I have thoroughly Unit Tested the proof of concept of HI, the Dataset and Linear Motion model.

4.2.3 UML

I have included the package diagram that describes the high level design of the project as well as the class diagram of the HI demonstrator.

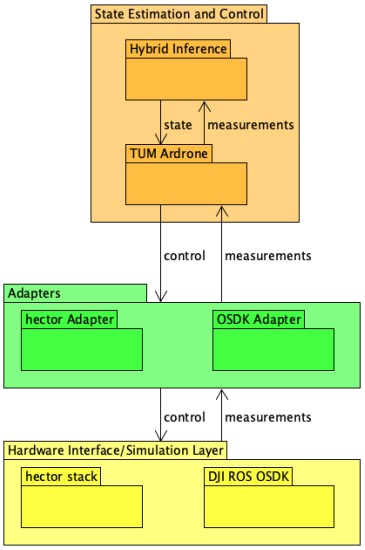

Figure 4.1: Package diagram of the project.

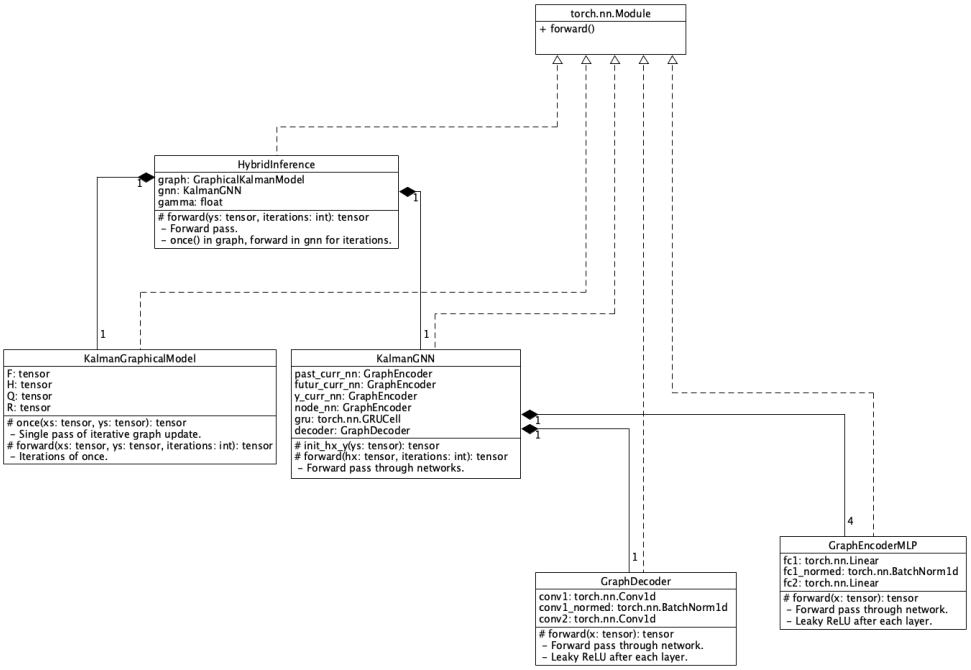

Figure 4.2: Class diagram for Hybrid Inference

Chapter 5: **Background Theory**

5.1 Literature Survey

5.2 Gazebo and ROS

In order to demostrate the effectiveness of the code that I have written it is essential that it is demonstrated first in a simulated environment. This neccesitates using Gazebo as it is the industry standard robot simulation software. In a similar token the industry standard robot development framework is ROS, which works with Gazebo and so this is the primary technology stack that I am using.

Both ROS and Gazebo are very powerful but they don't have a particularly easy onboarding process. For learning basic ROS concepts I used a service called Robot Ignite Academy that has some simple tutorials to make the process of learning the ROS way of doing things a bit quicker. One challenge I had at the very early stage was understanding what ROS actually was. There are very few high level descriptions of it, and they are certainly not the first topic on the ROS tutorials pages.

What I learned was that ROS is a paradigm of programming robots as well as an implementation of that paradigm. The paradigm is that each peice of code on the robot is a Node that takes data from a Topic, processes it and outputs data for other Nodes to use on a Topic. Some Nodes don't do both of these and some also directly effect change in the robot, think of a Node that controls a motor, it would read and change the voltage to the motor.

The other concepts that make up ROS are that of Services which are synchronous and Activities that are asynchronous, both operate on the Client - Server pattern. Topics as I mentioned are communication channels that operate on the Publisher-Subscriber pattern. And Messages which actually have quite a bit of intricacy and can be quite confusing. There are 3 types of Message. Regular Messages that are published to Topics. Then Actions and Services also define their own formats of messages. They will often use standard messages defined for use in Topics but additional message formats will be generated for each Action or Service.

5.3 Monocular SLAM with scale recovery.

In two related papers [10][11], Engel et al. present a technique of Monocular SLAM that is able to recover absolute scale. In order to understand why this is significant and important some background is needed.

The primary problem of visual SLAM techniques is that of creating a high quality depth map of the surroundings. This can be accomplished by stereo vision, which is where two cameras are used and the distance and rotation between the two of them are known. Ideally the axes are aligned and there is no rotation between the cameras though this can vary depending on the specific requirements. This can be extended to use three or more cameras but in the more general case, known as structure from motion, it is possible to use a single camera. In that case we are trying to recover the transformation and rotation between frames. This is

possible however it is not possible to recover absolute scale from this technique as we don't know the absolute distances between frames. In the stereo case the information about the position and orientation of the cameras relative to each other is known a priori and so we can recover absolute scale.

Engel et al. tackle the issue of scale recovery by using data from the IMU which usually consists of 3 accelerometers and 3 gyroscopes and often a magnetometer or other absolute scale sensors such as altimeter or barometer. This is standard equipment on quadcopters as it is needed to maintain stability. They use the information provided by this sensor in conjunction with the structure from motion equations in order to recover the absolute scale. In order to estimate the scale factor which is usually referred to as λ they take a maximum likelihood approach, which in simple terms means they estimate the λ that maximises the likelihood of the x and y positions measured by onboard sensors. In order to solve successfully they turn it into the negative log likelihood which you then minimize. This is a common trick and in this case it leads to a closed form solution for λ. This is important as it reduces the computational cost and makes it more feasible with onboard computation. In the paper they use a Parrot AR drone which has very constrained payload and computation capacity so they use offboard computation.

In the second paper [10] they present the full system including the scale estimation and demonstrate its effectiveness in position flying and holding.

5.4 State Estimation

The core method used for state estimation in the face of noisy sensors is the Kalman Filter. It is used by Engel et al.[11] for state estimation and sensor fusion which are its most common uses in this field.

Let us formally define the problem. The system we are measuring is assumed to be characterised by a Hidden Markov Model, this is how Kalman characterised the dynamical system we are interested in [15]. In this formulation there are hidden states x and observations y. The system being a Hidden Markov Process means that it is characterised by the following equations:

$$x_t = Ax_{t-1} + Q_t \tag{5.1}$$
$$y_t = Hx_t + R_t \tag{5.2}$$

Where A is the transition matrix from time $t-1$ to t, and ξ_t is the noise at time t. This represents that the model dynamics are stationary so A is fixed but there is noise in transitions, that is transitioning between states is non-deterministic. H is the measurement matrix and R_t is the noise in the measurements. This models the reality of noisy measurements that may not be correct and in fact represents the true problem. We want to recover the true measurements despite being given noisy observations.

The Kalman Filter gives an optimum estimate of x which I will call x^* by deriving three matrices Φ^*, P^* and Δ^*:

$$\Delta^*(t) = A_{t+1;t} P_t^* H_t^T [H_t P_t^* H_t^T]^{-1} \tag{5.3}$$
$$\Phi_{t+1;t}^* = A_{t+1;t} - \Delta_{t+1;t}^* Ht \tag{5.4}$$
$$P_{t+1}^* = \Phi_{t+1;t}^* P_t^* A_{t+1;t}^T + Qt \tag{5.5}$$

Note that in the Kalman formulation he also considers non stationary dynamic systems. In our situation we assume stationarity which allows us to drop the time specification on the transition matrices A and H.

With these matrices, the optimal estimate of state at time $t+1$, $\hat{x}_{t+1|t}$ is given by

$$\hat{x}_{t+1|t} = A^* \hat{x}_{t|t-1} + \Delta_t^* y_t \tag{5.6}$$

The estimation error x' and covariance of the error, cov x' are

$$x'_{t+1|t} = A^* x'_{t|t-1} + Q_t \tag{5.7}$$
$$\text{cov } x' = P_t^* \tag{5.8}$$

From this we can see that the Kalman Filter is an iterative process where each iteration builds upon the previous best estimate of state. To recover the estimates of the true measurements we simply need to multiply the best estimates of the state by the measurement matris H.

Kalman and Bucy extended the Kalman filter that, in the form stated above works for discrete time, to the continuous case the year after the original paper [16].

These equations only apply to linear dynamic models which is something of an issue given that most real life applications are of non linear dynamic systems. To solve this we use Taylor expansions to linearise our non-linear models around the current state [28]. This does unfortunately add a large overhead as we need to re linearise at each time step to avoid accumulating linearisation errors.

5.5 Graph Neural Networks

As the technique I am implementing makes use of Graph Neural Networks (GNNs) I will first explore what GNNs in relation to regular neural networks (NNs).

5.5.1 Neural Networks

To understand GNNs it is first necessary to understand some key properties of NNs. The first is that NNs are Universal Function Approximators [14] That is given sufficient hidden layer neurons a neural network can approximate any real valued function no matter how complex. This is the basis for their power and flexibility.

This doesn't come without drawbacks however. The cost of this flexibility is that training a universal function approximator (or a feed forward network with an arbitrarily large hidden

layer) is not necessarily feasible in computational terms or in terms of the amount of data required.

The paradigm of learning that NNs ascribe to is called representation learning. That is learning meaningful representations of data in order to achieve some useful outcome.

In order to overcome this limitation there has been great progress in formulating specific architectures of neural networks. Some good examples of these are Convolutional Neural Networks (CNNs) and Residual Networks (ResNets). One of the keys to these models successes is that they constrain the neural architecture. For these two in particular they constrain the model to Euclidean space and particular dimensions within this space. This restriction means that the resulting network is no longer a universal function approximator but that also reduces the amount that the network has to learn like about the rules of Euclidean space and what a dimension within the data is. It is precisely these restrictions that enable such networks to learn with fewer data samples and to a greater accuracy than would be possible otherwise.

This can be rephrased with a slightly different interpretation by seeing the constraints on the network as introducing inductive biases into the model. This biases it to learn certain features of the dataset and not others.

5.5.2 Graphs

We define graphs as a collection of nodes (also known as vertices though I will use the word node from now on) and a collection of edges over those nodes. Graphs are prevalent across many different areas of science and are a particular focus of many Computer Scientists. Famous examples of graph algorithms include Djistra's Algorithm, Depth First Search and A star. Why are graphs so prevalent? Well the answer is that they have great expressive power to represent objects or concepts and their relations.

The flexibility of graphs is mirrored in the great variety of operations that we might want to carry out over them. At the same time there is a huge number of different configurations of graphs varying over degree of connectivity of each node, whether edges are weighted or directed, containing cycles or not and whether the graph changes at all. Whether the edges represent different kinds of relation and what information each node contains.

5.5.3 Graph Neural Networks

We consider each node to have a vector label or feature. This may contain information about edges or not. If we were to try and apply CNNs or RNNs to graphs in this form, we could do so by stacking the node features and operating over that. In this case the node features would contain the information about edges.

This is not ideal because it enforces or biases the model to a particular ordering of nodes where there might not be one. We could overcome this by computing over all possible orderings of nodes. This would however add significant computation. [32] It also prevents the model from exploiting the graphs inherent structure, as it had been folded into the nodes themselves. It is clear then that this is not the best approach.

Original GNN

Now I will explain the original GNN proposed by [27]. The following explanation is drawn from [32] with some rewording. The goal of the GNN is to learn an embedding $h_x \in \mathbb{R}_s$ of

each node x which contains the information from the node and its locality. This embedding is then used to generate an output o_x of each node, often called the decoding step, where the embedding of each node generates a prediction of some kind for example the node labels.

In order to compute these embeddings and outputs we define two parametric functions, the first is f a local transition function which is shared by all nodes, and g the local output function. These are defined as:

$$\mathbf{h}_v = f\left(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]}\right) \tag{5.9}$$

$$\mathbf{o}_v = g\left(\mathbf{h}_v, \mathbf{x}_v\right) \tag{5.10}$$

where $\mathbf{x}_v$, $\mathbf{x}_{co[v]}$, $\mathbf{h}_{ne[v]}$, $\mathbf{x}_{ne[v]}$ are the features of v, the features of its edges, the states, and the features of the nodes in the neighborhood of v, respectively.

This can be condensed by stacking vectors to form matrices. Then we have $\mathbf{X}$, $\mathbf{X}_N$, $\mathbf{O}$ and $\mathbf{H}$ consisting of all features, the features of the nodes, the outputs and the states respectively. In this form we then have global transition function F and global output function G.

$$\mathbf{H} = F(\mathbf{H}, \mathbf{X}) \tag{5.11}$$

$$\mathbf{O} = G(\mathbf{H}, \mathbf{X}_N) \tag{5.12}$$

$$\tag{5.13}$$

The GNN takes inspiration from Banach's fixed point theorem [17] and uses the following iterative scheme to compute the states:

$$\mathbf{H}^{t+1} = F(\mathbf{H}^t, \mathbf{X}) \tag{5.14}$$

This converges to a stable H exponentially fast for any start point of $\mathbf{H}_0$

We use NNs for the parameterised functions f and g.

As stated in [20] the model is trained using the Almeida-Pineda algorithm [6][21] where the states are computed to convergence at which point the output and losses are computed. The gradients of the weights in the transition function and output fuction networks are computed from this final state and the weights updated according to the optimisation algorithm selected.

Recurrence

An extension or variation of the GNN is the Gated Graph Neural Network (GGNN). The key reason for the introduction of the GGNN is to reduce the restrictions on the transition function. Rewording what was stated in [20] the paper that proposes the GGNN they outline the challenge of the original GNNs learning scheme. As I just described, in the regular GNN the loss and gradients are computed relative to the final converged state. This requires that the parameters of the NNs be constrained so that it is a contraction map, otherwise convergence cannot be guaranteed. In order to remove this restriction, as there is evidence that long range dependencies ar lost in the contraction map scheme, they add a Gated Recurrent Unit (GRU) [8]. In this situation information is conserved at each time step and

while gradients are still computed from the final state and output, the GRU is unrolled backpropagated through time (BPTT) [21] is used to propagate these gradients across all time steps of the iterative procedure. This removes the constraint on the parameters of the component NNs.

MPNN

After the introduction of the GNN there have been many additional variants have been proposed such as Spectral Networks and Graph Convolutional Networks (GCN) and Graph Attention Networks.

There have been proposed some frameworks that generalise these various variants of GNNs. The framework relevant to this application is called the Message Passing Neural Network (MPNN) which was proposed by [13].

Again I am using the explanation of [32] with some rewording as it is an excellent high level explanation.

The MPNN has two phases, the message passing phase and the readout phase. The message passing phase runs for T steps and there are two parts the message function M_t and the update function U_t, $\mathbf{e}_{vw}$ is the edge feature on the edge from v to w.

$$\mathbf{m}_v^{t+1} = \sum_{w \in N_v} M_t(\mathbf{h}_v^t, \mathbf{h}_e^t, \mathbf{e}_{vw}^t) \tag{5.15}$$

$$\mathbf{h}_v^{t+1} = U_t(\mathbf{h}_v^t, \mathbf{m}_v^t) \tag{5.16}$$

The readout phase computes a vector for the whole graph using the readout function R and T denotes the total number of time steps or iterations.

$$\hat{\mathbf{y}} = R(\mathbf{h}_v^T | v \in G) \tag{5.17}$$

This is modifiable of course to allow a per node output. In this case the readout function would be similar to the output function of the original GNN. In fact there are many parallels between the original GNN and the MPNN but the MPNN has more flexibility which allows it to capture the majority if not all of the supervised variants of the GNN.

We can also implement the GGNN using this framework.

5.6 Hybrid Inference

Now I introduce the technique that form the core objective of this project, Hybrid Inference. First introduced in [26] which also gave examples on Lorenz attractors and Kalman filters. While Kalman Filters give optimal estimates in the face of noise, it is almost impossible for them to be completely precise due to that noise. This we can see by observing the error in the estimates at each time step.

Absent an accurate fix of state, ie. a noiseless measurement, the error grows continuously to the point where estimates may no longer be meaningful.

At the same time it is now possible to train a neural network to directly estimate position given noisy inputs [31]. The concept of Hybrid Inference is to leverage the relative strengths of pre-existing knowledge, in this case the Kalman Filter that accurately describes the behaviour of linear or linearised dynamic models up to a degree of error, and Deep Learning techniques that are able to model highly non linear systems but require large amounts of data to train. In the Hybrid Inference model, expert knowledge is incorporated by integrating the model of the system with a Graphical Neural Network (GNN) modelling the residual error to improve accuracy above what is possible with a Kalman filter alone.

Reexamining the problem solved by the Kalman filter, Satorras, Akata and Welling re-formulate the problem to be a maximum likelihood problem[26]. Where the states are $\mathbf{x} = \{x_0, x_1 ... x_k\}$ and the observations are $\mathbf{y} = \{y_0, y_1 ... y_k\}$ In this context, the task is to predict the optimal estimate of $\mathbf{x}$, $\hat{\mathbf{x}}$ which is defined as

$$\hat{\mathbf{x}} = \underset{x}{\operatorname{argmax}}\ p(\mathbf{x}|\mathbf{y}) \tag{5.18}$$

Again as we assume a Markov process and that the transition is stationary this can be expressed as

$$p(\mathbf{x}, \mathbf{y}) = p(x_0) \prod_{t=1}^{T} p(x_t|x_t - 1) p(y_t|x_t) \tag{5.19}$$

They model this as an iterative optimization process to arrive at $\hat{\mathbf{x}}$ Specifically they define a recursive update operation for the general case and then formulate it for the Hidden Markov case

$$x_t^{(i+1)} = x_t^{(i)} + \gamma M_t \tag{5.20}$$

Where M_t represents the sum of matrix products, which they call messages, from $x_t - 1$ to x_t from $x_t + 1$ to x_t and from y_t to x_t.

In our case these messages turn out to be

$$x_{t-1} \rightarrow x_t = -Q^{-1}(x_t - Fx_{t-1}) \tag{5.21}$$
$$x_{t+1} \rightarrow x_t = F^T Q^{-1}(x_{t+1} - Fx_t) \tag{5.22}$$
$$y_t \rightarrow x_t = H^T R^{-1}(y_t - Hx_t) \tag{5.23}$$

The graphical interpretation is of the xs and ys at each time step forming nodes in a graph. The edges are the same as the direction of the messages, that is from $x_{t-1} \rightarrow x_t$, $x_{t+1} \rightarrow x_t$ and $y_t \rightarrow x_t$. The messages passed over these edges iteratively update the xs and after some iterations they converge to a best estimate.

This graphical interpretation allows us to define an equivalent graph with different dimension nodes but the same edges. These are known as h_x and h_y which stands for hidden nodes relating to x and y respectively.

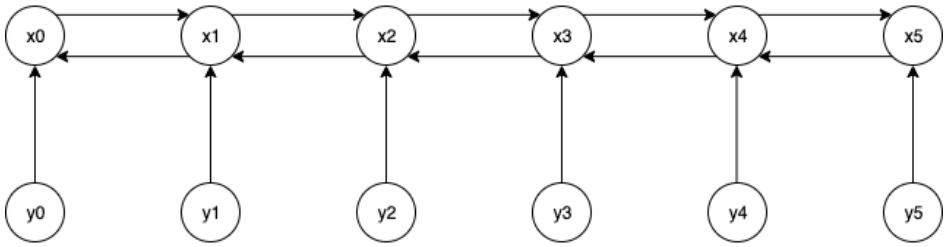

Figure 5.1:

The key part of the paper is to define a GNN that operates over this graph with a message passing routine over the nodes that connect edges and the messages passed in the original graph. More specifically for each type of edge, for example $y_t \to x_t$, we define a feedforward neural network that takes the source node, the target node and the message and outputs an encoding of the edge, I will refer to these as edge models. These encodings are summed according to their source node and then passed through a separate feedforward network which I will call the node model.

The output of this is passed through a GRU along with the previous h_x in order to produce a new estimate of h_x. The interpretation of this is that edge feedforward networks compute the residual error over the edges, the node model computes the residual error left in the h_x and the GRU allows some of the past residual error to propagate into the current estimate. The final step passes the new h_x through a decoding step to produce an additional corrective factor in addition to M_t called ϵ.

This gives us the final general recursive update rule

$$x_t^{(i+1)} = x_t^{(i)} + \gamma(M_t + \epsilon_t) \tag{5.24}$$

5.7 My Achievements

In the HI paper they implement a Kalman Smoother over linear and non linear systems however in both of these cases they didn't cover the case of having inputs to the system.

This is necessary in order to be useable in the context of drones as there are inputs required at every point, both to maintain stability as well to navigate or accomplish anything useful.

To add inputs I needed to change the probability distribution we are modelling.

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}, \mathbf{u}) \tag{5.25}$$

where $\mathbf{u}$ are the inputs.

This changes the recursive update function (Eq. 5 in [26]) to be

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \gamma \nabla_{\mathbf{x}^{(i)}} log(p(\mathbf{x}^{(i)}|\mathbf{y}, \mathbf{u})) \tag{5.26}$$

Which in turn changes the derived input messages to be

$$\mu^{(i)}_{x_{k-1}\to x_k} = \frac{\partial}{\partial x^{(i)}_k} log(p(x^{(i)}_k | x^{(i)}_{k-1}, u_k) \tag{5.27}$$

$$\mu^{(i)}_{x_{k+1}\to x_k} = \frac{\partial}{\partial x^{(i)}_k} log(p(x^{(i)}_{k+1} | x^{(i)}_k, u_k) \tag{5.28}$$

$$\mu^{(i)}_{x_{y_k}\to x_k} = \frac{\partial}{\partial x^{(i)}_k} log(p(y_k | x^{(i)}_k) \tag{5.29}$$

Formulated for the Kalman filter model this gives us

$$\mu^{(i)}_{x_{k-1}\to x_k} = -\mathbf{Q}^{-1}(x_k - (\mathbf{F}x_{k-1} + \mathbf{G}u_k)) \tag{5.30}$$

$$\mu^{(i)}_{x_{k+1}\to x_k} = \mathbf{F}^T\mathbf{Q}^{-1}(x_{k+1} - (\mathbf{x}_k + \mathbf{G}u_{k+1})) \tag{5.31}$$

$$\mu^{(i)}_{x_{y_k}\to x_k} = \mathbf{H}^T\mathbf{R}^{-1}(y_k - \mathbf{H}x_k) \tag{5.32}$$

Where $\mathbf{G}$ is the gain matrix over the inputs u.

In a wider context this is a small but significant extension that shows that this technique could be used in the full spectrum of applications of the Kalman Smoother.

I detail the implementation in Work Completed and share the results which are comparable with the original HI implementation.

Chapter 6: **Work Completed**

6.1 ROS and Gazebo

The first task to carry out was to install ROS and Gazebo. As mentioned earlier ROS is the framework into which my code will fit. Gazebo is a simulation tool that I will be using to verify everything before deploying anything in reality. Luckily Gazebo is included with ROS (with some exceptions) so installing it separately is not necessary. If you need the most recent version of Gazebo this is not true, you need to follow some additional steps to install it and link it with ROS so they interact correctly.

I did have some confusion about which version of ROS I needed to be compatible with various things but I made the decision to stick with the most recent version of ROS and the standard version of Gazebo packaged with that. This simplifies the setup for anyone seeking to reproduce my project.

The ROS website[1] has an excellent guide on installing ROS for the first time on Ubuntu and I highly recommend it.

After that I installed the hector stack. This was quite a long process of trial and error as there does not seem to be a guide for getting started with this set of packages. I first tried to install them from apt as they are available there. That did not work and I believe the issue is that they were built for an earlier version of ROS but packaged for melodic without any changes. Regardless of the reason that did not work. I was trying to install the hector quadrotor and hector gazebo packages as those were the only packages that I believed that I needed. I tried to clone them into the src folder of my catkin workspace. While trying to build it failed because of dependencies on other hector packages, namely hector slam hector models and hector localization. Then it was missing the geographic mesages. Then it failed because it was missing qt4. I installed both of those with apt. The full process took a while longer than this explanation as you only find out anpther dependency is missing after building again. A final issue was that the memory usage at certain points spikes. It turned out that this was causing the VM to run out of memory and stop compliation. This caused very confusing error messages as they didnt seem to be for anything in particular. I finally figured out with the help of htop and allocated more memory to the VM which solved the issue. One anachonism of this stack is that it doesn't seem to build in the right order. It will often fail only to complete more after rerunning. At one point it was necessary to continually rerun catkin_make about 7 or 8 times in a row. The only cause for concern is when it stops at the same percentage built more than once.

6.2 Hardware for Demonstration

My original plan was to use a Raspberry Pi for onboard computation when I transition to operating on the physical drone rather than in Gazebo. This was for a couple of reasons, primary was the low power draw while still offering gigahertz computation. I have a Raspberry Pi 2 and I planned to purchase a Raspberry Pi 4 for the final implementation due to its increased compute power.

While experimenting with my RasPi 2 I found some limitations with its implementation of Python as well as concerns about its prospective performance due to reports found online. I

researched alternatives, as there are a wide variety of Single Board Computers on the market now. While looking I found that NVIDIA produce the Jetson Nano and specifically the SDK kit, which is very similar to the RasPi but has native support for PyTorch, the deep learning library I am using as well as having 128 GPU cores. This enables me to take advantage of parallelised matrix computation with the associated performance improvements. On top of this it only consumes up to 10W of power, admittedly this is double the 5W draw of the RasPi but 10 is the maximum, not necessarily what it will consume. I will explore power consumption in the second part of the project.

After the Nano arrived I installed PyTorch and ROS onto it. This allowed me to verify the steps needed for installing both packages and ensure the most streamlined set of instructions.

There were some issues with the original version of PyTorch installed due to the Nano using an ARM CPU. This means that it has the aarch64 architecture and so compiled executables are trickier to get hold of and often are not quite the same as x86. In this case PyTorch uses some external libraries for some of the matrix operations, these include OpenBLAS and MKL. In the first version I installed these were not present, preventing matrix inverse operations. This is critical to both Graphical Kalman Smoother and by extension the HI implementation.

I tried a number of solutions including installing from source however in the mean time an updated version of PyTorch was released for the Nano which included OpenBLAS solving the problem.

In the end there were a number of factors that prevented the eventual hardware integration.

6.3 TUM code

The first step in working with the TUM code was to compile it. I was aware that it was written to target ROS Fuerte with some modifications to support Indigo which is 3 versions behind Melodic.

I first attempted to build the project on the Jetson Nano as I was planning to use it as my development platform as well as the deployment environment if possible. This was not possible. The build failed due to a dependancy, ardrone_autonomy. This is a wrapper for the Parrot AR drones SDK. It also provides a number of message definitions. I attemped first to build the dependancy on the Nano but there are issues with compiling that code on an ARM platform. This prompted me to drop the dependancy and the speed of compilation prompted me to abandon the Nano for development.

I found that the only code from the ardrone_autonomy package were the messages. I then extracted the message definitions from the ardrone_autonomy package into my own package.

After solving that issue I ran into much more serious issues with a third party library called libcvd which is an OpenCV alternative that PTAM, the monocular SLAM component. This package is packaged as a thirdparty library inside a tarball with two other libraries. Upon building it would fail on apparently legal C++ code. I tried to use the most recent version of libcvd, pulling it from github, building and installing it on the system separately. This didn't work so I tried placing it into the thirdparty tarball. That also did not work. I found that the directory structure of the new version did not match the original and there was a Makefile I needed to replicate. I then found that there were some files that had been deprecated but the TUM code relied upon so I pulled them over from an old version of the library. This on it's own did not fix the issue but when I found a final Makefile I could add the files I had pulled over into the list of artifacts that would be made available by that stage of the build.

Then I found some namespace issues as another dependency was not being found in the expected manner. I had to set a definition in order to fix that problem. I also had to remove the GUI section of the code as it had more serious persistent issues. If I have time spare after completing the project I will go back and try and fix that as well. Finally there was a section of code that relied upon 'tf', which is a geometry package of ROS but is now deprecated. I had to rewrite that section to use 'tf2' which is the replacement for 'tf'.

After building it ran without issue.

I then explored how to get it to interact with the hector stack. I looked through the hector stack and it is very large. A lot of it is not needed for my project so I considered extracting the core functionality that I want, create a world with a quadcopter that acts correctly. Unfortunately the hector stack has a large number of interdependencies so it is infeasible to extract only certain components, at least not at this stage. I have identified where they will interface. They use different conventions for positions so I need to create an adapter.

6.4 Hybrid Inference

6.4.1 Data

A prerequisite for any task that includes neural network is collecting or creating the data for it to train on. In this case my dataset is created as at this stage I am firstly verifying that the technique works as expected and I can implement it. After this stage I will formulate the real problem and collect data in Gazebo for that stage.

My dataset consists of position data generated by a simple linear model with Gaussian noise in the transition matrix as well as the measurement matrix. This is very similar to one of the experiments carried out in [26]. This is intentional in order to have comparison data, though the model I am using is likely to be much simpler than that used in the paper.

The testing strategy I have used for the Linear Model and the Dataset is designed to verify the core performance. That the amount of data created is correct, that the various options operate as expected and that the output data is in the format expected.

6.4.2 Graphical Model, GNN and Hybrid Inference

My original understanding of the Hybrid Inference formulation was that the graphical model was a regular Kalman Filter and the GNN was formulated in a similar fashion. As I describe in the Background Theory section that is not the case. The graphical model is a reformulation of the problem and solution. I realised this after reading the paper with reference to the GitHub repository with their implementation [25]. It took a fairly long time to understand their code as it is not really documented and has a somewhat confusing structure.

Implementing the graphical model was then relatively straightforward though trying to keep it self contained and modular complicated things slightly. My implementation of the GNN is quite problem specific. At the moment I can't see much way to make it more generic. Given that I will have to reformulate for the reasons just stated I will explore where the commonalities are and whether a generic version is feasible or desireable. It differs quite a lot from the Satorras' formulation mainly due to my desire to improve the interpretability of the code and give a cleaner flow and structure. Functionally they are equivalent except that

my node model is a 3 layer feedforward network whereas the one they implement is 2 layers. I also decided not to implement different modes for graphical only or GNN only as I specified these as separate standalone, or semi standalone in the case of the GNN, classes. As I had implemented the two components separately it made the Hybrid Inference class very simple as it is just a composition of the two.

In the second term I focused initially on the HI section of the project. I first sought to verify the results of the HI paper before continuing with solutions to the challenges identified at the end of the first term. Namely that of inputs and that of prediction.

Firstly I compared the HI model with the graphical formulation of the Kalman Smoother. Initially the results were significantly worse which is not supposed to ever happen. This triggered an in depth review of my implementation to find the source of the error. Every line was stepped through and evaluated, my initial idea was that the GNN part of the implementation was the most likely culprit. I finally discovered that the cause was the number of iterations that the HI implementation carries out before returning the solution. It was 50 compared with the Graphical Kalman Smoother using 200. Once they were equalised the HI performed better.

6.5 Self Evaluation

I would consider the project to have gone reasonably well. I certainly had plenty of challenges and some scaling back of ambition but I am happy with the final results that I was able to achieve.

The next stage of this project will be to actually integrate it with some hardware. Due to the complications encountered I had to scale back my ambition to remain at the simulation level.

In terms of process, I would say that my consistency of work was good, I managed my time pretty well in relation to other commitments balanced with this one to ensure that there was some measure of progress each week. In terms of individual decisions, I think that the decision to focus on the Hybrid Inference technique was a good one in terms of enabling me to challenge myself, learn a large amount about new areas of theory, notably in Hidden Markov Models, Kalman Filters and Graph Neural Networks.

I would say that I did a few things wrong. The first was not doing enough research before the project started. I should have spent more time checking for all the resources I could use. A key example of this was the Monocular SLAM project I wanted to replicate. I found that it was open source and more generic than I had assumed so I could reuse it. This led to me pivoting my project aims to focus more on Hybrid Inference. Following on from that, I was far too ambitious in my original planning. I had planned to implement two serious papers techniques, integrate them in Gazebo and then also on the DJI Matrice. This was far too optimistic as was evidenced by the change in project aims and scaling back of ambition.

The key takeaway for me is the limitations of the Waterfall method of project planning and management. This project was structured as some version of Waterfall in that we planned the deliverables, milestones, timing and essentially every part of the project right at the start of the project.

This has some advantages in that it forces you to spend some time thinking about the project which should reduce the amount of surprises. The downside is that many things change during the course of a project of this size and duration. Particularly for those of us working with

entirely new technologies in different application areas than we have worked on before, ie. not some kind of web app or desktop/mobile application, it is very hard to estimate the amount of time needed for most of the tasks we were planning. Unsurprisingly many ended up taking much longer than estimated.

I think that preplanning is necessary but the extent and detail must vary depending on the extent of familiarity with the domain, tools and task to be carried out. In any case there will often be regular course checks and this is where the Agile approach has significant advantages in setting short term goals that are better understood and more controllable over large goals over longer time periods where the scope for misunderstanding is much greater.

The other thing that I have learned about doing a project is that fixing issues generally takes much more time than implementing new functionality. At the same time it is much harder to plan your time when fixing issues as there is no reasonable way to estimate when you will understand the issue enough to solve it. This is much more of a problem when using little used tools or technologies. For the majority of application development there is a large amount of knowledge available online to refer to when encountering difficulties. Unfortunately for me ROS and Gazebo have more limited communities using them. And when implementing new techniques, there is no reference other than the paper, and maybe source code if made available via GitHub.

Bibliography

[1] melodic/Installation/Ubuntu - ROS Wiki. http://wiki.ros.org/melodic/Installation/Ubuntu.

[2] An on-device deep neural network for face detection - apple. https://machinelearning.apple.com/2017/11/16/face-detection.html, Nov 2017.

[3] Caracas drone attack, Jan 2020.

[4] Gatwick airport drone incident, Feb 2020.

[5] Alenyà, G., Martínez, E., and Torras, C. Fusing visual and inertial sensing to recover robot ego-motion. *J. Field Robotics 21* (01 2004), 23–32.

[6] Almeida, L. B. *A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment*. IEEE Press, 1990, p. 102–111.

[7] caochao39. caochao39 on github hku_m100_gazebo. https://github.com/caochao39/hku_m100_gazebo.

[8] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR abs/1412.3555* (2014).

[9] DJI. Onboard sdk overview. https://developer.dji.com/onboard-sdk/.

[10] Engel, J., Sturm, J., and Cremers, D. Accurate figure flying with a quadrocopter using onboard visual and inertial sensing. *IMU 320* (01 2012).

[11] Engel, J., Sturm, J., and Cremers, D. Camera-based navigation of a low-cost quadrocopter. pp. 2815–2821.

[12] Geffner, H., and Bonet, B. *A Concise Introduction to Models and Methods for Automated Planning: Synthesis Lectures on Artificial Intelligence and Machine Learning*, 1st ed. Morgan & Claypool Publishers, 2013.

[13] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *CoRR abs/1704.01212* (2017).

[14] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 5 (1989), 359 – 366.

[15] Kálmán, R. E. A new approach to linear filtering and prediction.

[16] Kálmán, R. E., and Bucy, R. S. New results in linear filtering and prediction theory.

[17] Khamsi, M. A., and Kirk, W. A. *An introduction to metric spaces and fixed point theory*, vol. 53. John Wiley & Sons, 2011.

[18] Kirkpatrick, D. D. The affordability of defence equipment. *The RUSI Journal 142*, 3 (1997), 58–80.

[19] Kose, T. Implementing dji m100 emulator in ros-gazebo for hardware in the loop. https://medium.com/@tahsincankose/implementing-dji-m100-emulator-in-ros-gazebo-for-hitl-e79f4bb077f6, May 2019.

[20] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. Gated graph sequence neural networks. *CoRR abs/1511.05493* (2016).

[21] Pineda, F. J. Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett. 59* (Nov 1987), 2229–2232.

[22] Putzky, P., and Welling, M. Recurrent inference machines for solving inverse problems. *CoRR abs/1706.04008* (2017).

[23] Romero, A., Kahou, S. E., Montréal, P., Bengio, Y., Montréal, U. D., Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. In *in International Conference on Learning Representations (ICLR* (2015).

[24] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[25] Satorras, V. G. vgsattorras on github hybrid-inference. `https://github.com/vgsatorras/hybrid-inference`.

[26] Satorras, V. G., Akata, Z., and Welling, M. Combining generative and discriminative models for hybrid inference. *ArXiv abs/1906.02547* (2019).

[27] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks 20*, 1 (Jan 2009), 61–80.

[28] Smith, G. L., Schmidt, S. F., and McGee, L. A. Application of state-space methods to navigation problems. NASA Technical Report R-135, 1962.

[29] Viet, C., and Marshall, I. Vision-based obstacle avoidance for a small, low-cost robot. pp. 275–279.

[30] von Stumberg, L., Usenko, V. C., Engel, J., Stückler, J., and Cremers, D. Autonomous exploration with a low-cost quadrocopter using semi-dense monocular SLAM. *CoRR abs/1609.07835* (2016).

[31] Yadaiah, N., and Sowmya, G. Neural network based state estimation of dynamical systems. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings* (July 2006), pp. 1042–1049.

[32] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., and Sun, M. Graph neural networks: A review of methods and applications. *CoRR abs/1812.08434* (2018).

[33] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., and Sun, M. Graph neural networks: A review of methods and applications. *CoRR abs/1812.08434* (2018).

Appendices