Chapter 1

Algorithm Design and Implementation

1.1 UKF Formulation

We begin by defining the following vector quantities:

$$tk : p, q, v, \omega, a \tag{1.1}$$

tk: define n and m

We then define the state vector $\mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{x} \triangleq \{\mathbf{p}, \mathbf{q}, \mathbf{v}, \omega, \mathbf{a},\}^T. \tag{1.2}$$

The covariance associated with each state $\mathbf{x}$ is a matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$. A set of $2n+1$ sigma points is then derived from $\mathbf{x}$ and $\mathbf{P}$:

$$\begin{aligned}
\chi^0 &= \mathbf{x} \\
\chi^i &= \mathbf{x} + \left(\sqrt{(n+\lambda)\,\mathbf{P}}\right)_i, \quad i = 1, \ldots, n \\
\chi^i &= \mathbf{x} - \left(\sqrt{(n+\lambda)\,\mathbf{P}}\right)_{i-n}, \quad i = n+1, \ldots, 2n
\end{aligned} \tag{1.3}$$

where λ is defined in terms of the following constants:

$$tk : \alpha, \beta, \kappa, \lambda \tag{1.4}$$

tk: TO BE CONT'D.

1.2 Software Design Considerations

Much of the impetus for creating the `kalman_sense` package came from a desire to create a generic UKF framework for estimating the state of an arbitrary system using any number of relative and absolute sensors. To achieve this, the `kalman_sense` package is organized in an object-oriented manner around an overarching abstract class called `UnscentedKf`. This abstract class contains a number of methods performing the different mathematical operations defined in Section **??**. These methods have been written in a generic manner in order to enable easy extension of `UnscentedKf` by other subclasses containing concrete implementations of various systems. Currently, the package contains exactly one such subclass, known as `QuadUkf`. This subclass contains methods and data structures related directly to estimating the state of a quadcopter or other rotorcraft UAV.

This object-oriented architecture is allows for a certain degree of system agnosticism. By this, we mean that the `UnscentedKf` class encapsulates the generic mathematics of the UKF without knowledge of particular system constraints. This class does little other than matrix mathematics and is designed to take as input the number of a system's states n and its number of sensors m. With this knowledge, `UnscentedKf` is able to populate a set of mean and covariance weights and intelligently perform all of the requisite linear algebra for the UKF formulation. All other knowledge of particular states, sensors, vehicle geometry, and other metrics is hidden within subclasses such as `QuadUkf`.

`UnscentedKf` behaves in a manner similar to a Java interface in that it requires the extending class to supply functions codifying a process model and an observation model for the system under scrutiny. These two functions, along with n and m, form the entirety of what `UnscentedKf` "knows about the vehicle." All other details, including the fact that the class is being used in a ROS environment, are hidden from `UnscentedKf`. It is worth noting that `UnscentedKf`'s only dependency is on the Eigen[1] C++ linear algebra library.

[1] `www.eigen.tuxfamily.org`

The subclass (`QuadUkf` for the remainder of this thesis) handles all of the ROS communications for the given system. Specifically, this class has callback functions for receiving sensor data and is responsible for publishing state and covariance estimates. The `kalman_sense` `main` method handles setup and teardown of the necessary ROS publisher and subscriber nodes.

Chapter 2

Experimental Design

2.1 Testing Considerations

Before venturing further, we should summarize the goals of the UKF framework described previously with particular attention to unmanned aircraft system (UAS) operations. This ROS package was designed with the express intent of producing estimates of the position vector $\mathbf{p}$ and orientation quaternion $\mathbf{q}$ of a rotorcraft UAV in real time. Thus, the experiments testing `kalman_sense`'s efficacy compare the filter's estimates of position and orientation to the "ground truth" as measured by a Vicon[1] motion capture system.

This system depends upon two sensors: a global-shutter camera and an IMU. The IMU used in this experiment contains a 3-axis accelerometer and 3-axis gyroscope (Figure **??**). To simulate both sensors moving through the scene in a manner reminiscent of hovering rotorcraft flight, a rolling test stand was constructed to carry the sensors safely throughout a large motion capture environment. Mounting the sensor suite on a large, steady, level platform allows for a high degree of control over the accelerations and angular velocities felt by the IMU, as well as the motion seen by the ventral camera. In order to validate the UKF framework's effectiveness under ideal conditions, a modern laptop computer with an Intel i7 processor and 16 GB of RAM was used for all computation.

[1] https://www.vicon.com

tk: figure of sensor mount

2.2 Materials

2.2.1 Computation and Sensing

1. One (1) MatrixVision mvBlueFOX-MLC Camera[2]
2. One (1) 1044_0 PhidgetSpatial Precision 3/3/3 High Resolution IMU[3]
3. One (1) Hewlett-Packard Spectre x360 Convertible Laptop 13-ac076nr[4]
4. Two (2) male Mini USB 2.0 to male USB Type A cables

2.2.2 Mobile Test Stand

1. One (1) Oklahoma Sound PRC200 Premium Presentation Cart[5]
2. One (1) 3D-printed sensor mount (see Figure **??**)
3. Two (2) 4-inch C-clamps
4. One (1) 1.2-meter 80/20 1515 rail[6]
5. One (1) 15 Series "L" Handle Linear Bearing Brake Kit[7]
6. One (1) tk: other rail screw
7. Two (2) tk: the part that goes inside the rail for both the brake and the screw
8. Three (3) 1-inch Vicon infrared retroreflector balls
9. Two (2) 0.5-inch Vicon infrared retroreflector balls

[2] https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html
[3] http://www.phidgets.com/products.php?product_id=1044
[4] http://store.hp.com/us/en/pdp/hp-spectre-x360---13-ac076nr
[5] http://www.oklahomasound.com/products/product-category/single/?prod=9
[6] https://8020.net/1515.html
[7] https://8020.net/6800.html

10. One (1) 0.7-meter length of $\frac{1}{8}$-inch-thick carbon fiber rod

2.3 The Experiments

A series of three experiments were designed to characterize the UKF framework's effectiveness in various regimes of motion. To characterize the accuracy of state estimates in lengthy, monodimensional motion, two "long walk" experiments were conducted—one in the x-direction, the other in the y-direction. These experiments were meant to determine changes in estimate accuracy over large, planar translations (for example, to uncover the evolution of error in the system over time while effectively manipulating only one state variable). For each long walk, the test stand was translated without rotation along the positive x- and y-axes over distances of approximately seven meters, then returned to the starting location via the same path.

The third experiment was a rectangular translation designed to characterize the the system's effectiveness when translated along two axes. Again, the cart was translated without rotation around the corners of a nearly square rectangle having sides approximately four meters in length.