MAD 1702

Plume Analysis and Detection

A Major Qualifying Project Report
Submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
in Aerospace Engineering

by

[illegible]

Eric Fast

[illegible]

Stephen Harnais

in Robotics Engineering

[illegible]

Ryan Wiesenberg

March 3, 2017

Approved by: [illegible]

Professor Michael A. Demetriou, Advisor

[illegible]

Professor Nikolaos A. Gatsonis, Coadvisor

[illegible]

Professor Raghvendra V. Cowlagi, Coadvisor
Aerospace Engineering Program, WPI

Abstract

This work involves the design and implementation of a gas-sensing mobile robot as an experimental tool to reconstruct a carbon dioxide plume in real-time based on concentration gradient and local wind speed measurements. The autonomous robot, an iRobot® Create 2, achieves navigation through an embedded micro-controller using on-board sensors and various sensor fusion methods. A mass flow controller and diffuser are used to dependably generate a plume that simulates a point source. A base station reconstructs the plume via a state estimator through data from the robot and transmits commands to guide it into spatial regions of interest. This method has applicability for unmanned vehicles tracking emissions of contaminants and their effects in the environment.

Acknowledgements

Thank you to the Aerospace Program at Worcester Polytechnic Institute for use of the lab space as well as the advising assistance of Professors Demetriou, Gatsonis, and Cowlagi. Also, a thank you to the Robotics Program at WPI for the technical assistance and consistent support throughout the process. In addition, appreciation is given to Joe St.Germain and Kevin Harrington for provisional use of sensors and other components. We would also like to extend our thanks to the employees at Toyota for the valuable feedback.

Contents

List of Figures

List of Tables

1 Introduction

In cases of the release of toxic or ecologically harmful gas, modeling the impact on the surrounding area is useful to mounting an effective response.[1] While there has been a great deal of research into gas source localization,[2, 3, 4] less work exists towards real-time reconstruction of entire gas plumes.[5] Direct analysis by humans may be dangerous, while simulations are blind to specific environmental conditions. Robotics could thus be employed in a mobile sensing platform to gather on-scene data relevant to reaction efforts.

A team at Worcester Polytechnic Institute (WPI) has developed a program which integrates the advection-diffusion equation to reconstruct and model a plume in the atmosphere using sensor data taken by an Wireless, Autonomous Aerial Navigator (WAAN) flying in the presence of the gas. Previous projects[6, 7, 8, 9] have begun the design process for a simulation using a Wireless, Autonomous Ground Navigator (WAGN) to evaluate the effectiveness of the algorithm in a laboratory. The current group seeks to continue the works of these projects and improve upon their design where possible.

This experimental setup involves the design of a WAGN which will report position and sensor data to a Computer for Autonomous System Execution (CASE). The CASE runs the computationally intensive task of estimating the plume based on a two-dimensional advection-diffusion model and a dynamic grid adaptation centered on the robot. The computer then returns navigation commands to direct the robot toward the area of highest interest in the concentration field. In addition, a gas emission system was designed generate the plume to be measured.

1.1 Previous Work

1.1.1 MAD1501

This team[6] conducted an analysis of preexisting robots for gas sensing and a review of potential technologies for our project to adopt. They selected and purchased a K-Team® Khepera IV for use as the ATV and developed preliminary code for navigation and guidance. They also designed the sensor mount and purchased the COZIR® gas concentration sensors still used by the project.

1.1.2 NAG1501

This team[7] created a preliminary design for the gas emission system, recommending components and analyzing a CFD model to prescribe a gas release rate that is detectable while staying below toxic levels. They also recommended the structural tubing that is now makes up the test stand for the system, and conducted a stability analysis to confirm its adequacy.

1.1.3 MAD1601

This team[8] further advanced the robot navigation system, creating a dynamic model for motion and linearizing so as to apply a Kalman Filter. They also machined the sensor mount proposed by MAD1501 and developed a method for collecting and exporting sensor data using an Arduino® Uno.

1.1.4 NAG1602

This team[9] reconfirmed the gas release rate proposed by NAG1501 with simulations of their own. They selected and purchased a tank regulator and mass flow controller for the gas emission system. They also proposed that the program LabVIEW be used to integrate all of the system components to simplify conducting the experiment.

2 Design Process

2.1 Computer for Autonomous System Execution (CASE)

The Computer for Autonomous System Execution (CASE) is the designated processing unit for all gas information received as well as the visual aide for the entire system, creating diagrams that show where the mobile unit is relative to its starting location for easy user interaction. This system turns the gas information into a map of the concentration gradient and then instructs the robot wirelessly how to move to the next area of interest.

2.1.1 Gas Simulation and Control

The CASE is responsible for processing the gas concentration data sent from the mobile unit and then uses it to command the robot's desired velocity and direction. The CASE configuration was built around several Figures of Merit (FOM): simplicity, versatility, and processing power. Because this experiment is just a stepping stone to a variety of different future tasks the CASE's method of processing the gas concentration information must be simple and versatile enough to take a system with only one ground unit with only one sensor on it all the way up to a several ground or air units with several sensors each. The processing power of the CASE is relatively unimportant as the calculations being done are relatively minimal and the most intensive process would be using a Universal Serial Bus (USB) camera to analyze the validity of an Extended Kalman Filter (EKF), the process of which is described in Section 2.3.

To fit these FOM, the CASE runs Ubuntu[10] 16.04 as it is very friendly for both experienced and novice programmers. Linux, the foundation of Ubuntu, can communicate easily with a wide array of peripheral devices needed to verify the EKF, analyze gas data received from the ground unit, and transmit communications to and from the navigators. The code used to communicate with the Wireless, Autonomous Ground Navigator (WAGN) and process the concentration information was written in Python[11] due to the language's ease of use and built in matrix functions which were used for coordinate system transformations as described in Section 2.3. As previously mentioned, the processing power required for the CASE was minimal and most modern computer workstations would suffice.

For the initial experiment, the program developed to help the robot navigate is very simple. It receives concentration and position information from the robot, determines the direction of strongest concentration and then gives the robot directions in the form of velocities in the x and y axis. This method would allow the robot to reach the plume directly. From there the robot would then be instructed to spiral outwards mapping out the surrounding area's concentration gradient.

2.1.2 Communication

The method of communication between the CASE and the navigators was chosen based upon several FOM: simplicity, versatility, reliability, range of use, and speed. Because this experiment is just a stepping stone to a variety of different future tasks simplicity and versatility are key in the selection of how the navigators interact with the CASE. This communication line has to also be incredibly reliable and cannot risk information loss or it will take a significantly longer period of time to survey an area, especially as the size of the area and resolution of the data increase. The range for this communication does not have to be very large however, as most use cases involve the CASE and navigators being in close relatively close proximity. For this experiment speed was a dynamically scalable factor with the frequency of sensor readings and relied entirely upon which gas sensors and WAGN were chosen, but near real-time simulation would be desirable.

To fit these FOM, a Wireless Local Area Network (WLAN) was selected because the network could be set up with an off-the-shelf wireless router has a much more reliable connection, larger workspace, and higher communication speed and bandwidth than Bluetooth®.[12] This network then allows the mobile unit and CASE to communicate via Transmission Control Protocol (TCP). TCP is not as quick as User Datagram Protocol (UDP) but is more reliable natively and is close to real time control.[13]

2.1.3 Kalman Filter Verification

To determine which Kalman Filter and sensor configuration, with or without the Inertial Measurement Unit (IMU), was more accurate, a verification method needed to be established. This verification method would act as a ground truth for the system, being able to determine the location of the robot anywhere within the experimental space without input from any other part of the Plume Analysis and Detection (PAD) system. For simplicity of the system, AprilTags[14] were used on top of all WAGNs and a camera was mounted to the ceiling of the experimental area to track their movements. The AprilTag system uses OpenCV[15], a computer vision library, to determine the pose of the top of the robot and then translate it into the robot's coordinate system, making the comparison between the Kalman Filter and the actual movement simple. AprilTags have over 200 different tags to be used simultaneously, allowing for future expansion. The CASE was responsible for the tracking and side-by-side comparison with the locational data streaming from the WAGN. For more information about the Kalman Filters, see Sections 2.3 and 2.4.

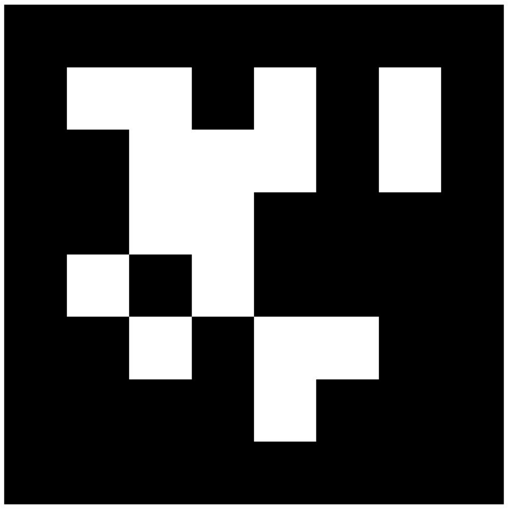


Figure 1: AprilTag.

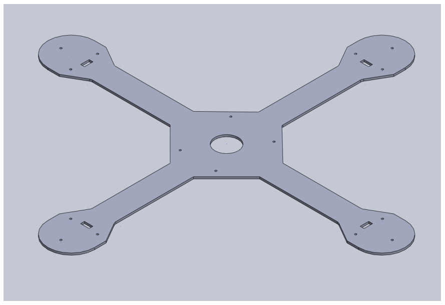
2.2 Wireless, Autonomous Ground Navigator (WAGN)

The mobile unit is a combination of two individual platforms, the sensing rig for PAD and the base which it is mounted upon, the WAGN, each of which has their own FOM. The PAD Instrument needs to be placed far from the ground to avoid floor disturbances. The WAGN is not only designed to be able to transport the instrumentation around a field but also collect the data, self-localize, and send its location and gas data back to the CASE.

2.2.1 Sensing

The previous iteration of this project designed the PAD Instrument on the left in Figure 2. An array of four CO_2 COZIR^{TM} sensors are arranged in a cross and positioned far enough from the ground to avoid the floor bias. One Modern Device® Rev.P anemometer was placed in the center of the PAD Instrument, just to the size of the center hole. However the thin metal plate tended to oscillate significantly and was relatively heavy because it was made out of Aluminum. To counteract this, our group made a more robust design for the PAD Instrument that included more support for the arms, made of Acrylonitrile butadiene styrene (ABS) plastic, as seen in Figure 2.

The CO_2 sensors function on the principle that different types of gas molecules absorb different wavelengths of light. By passing the light through an optical filter tuned to the absorption band for CO_2, concentration can be accurately determined from the drop of intensity taken at a maximum rate of 20Hz.


(a) Original.

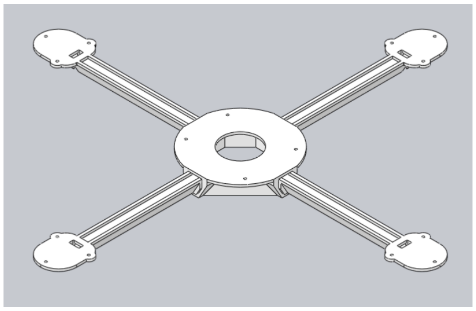

(b) New.

Figure 2: Plume Analysis and Detection Instrument.

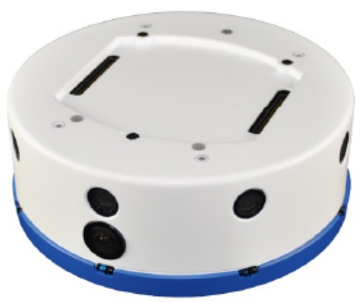
Local wind is measured using a thermal anemometer, which determines wind speed by measuring the change in temperature across a heated wire exposed to the airflow.

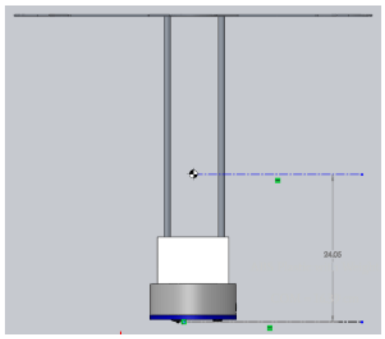
2.2.2 Robotics

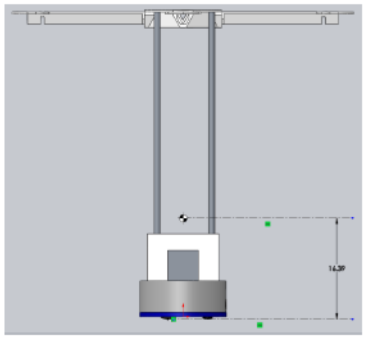
The previous project selected the K-Team KheperaIV[16] as the WAGN, which is a two-wheeled drop center robot with front facing array of long-range ultrasonic distance sensors, short-range omnidirectional Infrared (IR) distance sensors, encoders on the wheels, an IMU, and an embedded Linux controller. This unit was initially chosen because of its large array of sensors but its base made the PAD Instrument very unstable as the Center of Gravity (CG) of the system was very far from the ground as seen in Figure 3.

The team selected the iRobot® Create 2 as an alternative option for the WAGN because of its wider and heavier base as seen in Figure 3. This design is more conducive for the transportation of the PAD Instrument because the WAGN is less likely to tip over due to the high CG. However, the Create® 2 does not have an embedded Linux controller or many sensors apart from encoders on the wheels.

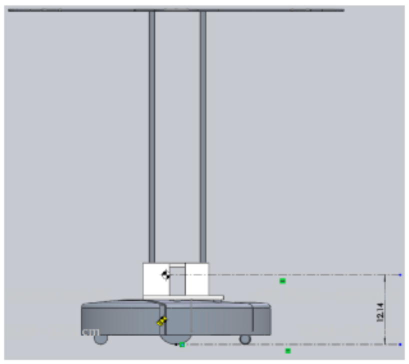
After reviewing both options, the Create® was selected as the optimum unit for the experiment. Although the Create® does not have nearly as many on-board sensors the PAD instrument is larger than the operational distance of the IR sensors on the Khepera. That means only the ultrasonics would have been used on the Khepera, which cannot be run simultaneously and are naturally unreliable. The Create® also does not have any native on-board processing but that is remedied with a separate processing unit, the selection of which is discussed in the following section. To account for the lack of additional sensors, a 9 Degree of Freedom (DOF) IMU was also mounted inside the WAGN. The most important factor in this decision was the Create®'s greater dynamic stability due to the lower CG and larger wheelbase allowing to move smoothly, giving more reliable localization.


(a) K-Team Khepera IV Base.


(b) With Original Mount.


(c) With New Mount.

(d) iRobot® Create 2 Base.


(e) With Original Mount.

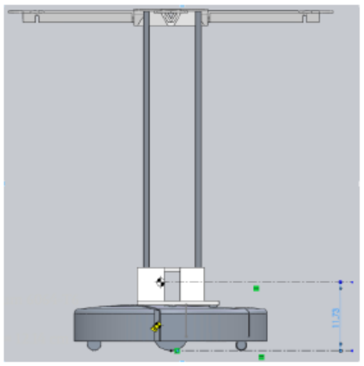

(f) With New Mount.

Figure 3: WAGN Design Comparison.

2.2.3 Computing

Because the WAGN does not have any on-board computing like the Khepera IV, a separate processing unit was needed. This unit was selected based on several FOM: simplicity, versatility, processing power, peripheral connections, and form factor. Because this experiment is just a stepping stone to a variety of different future tasks simplicity and versatility are key in how the robot receives its commands, processes them, and reports back its sensor information. The unit would also need enough processing power to handle the linear algebra associated with an EKF as well as receive incoming connections and process sensor data, all processes that are very computing intensive on microprocessors. As was mentioned in Section 2.1 the CASE and WAGN would communicate over a WLAN meaning the unit would need to have Wifi to establish a connection. Both the Create® [17] and the COZIR™ [18] sensors communicate over serial so the microprocessor would also need a way to interface with them all simultaneously. The Create® has a relatively small form factor and only a little room for storage so a laptop would be an inappropriate selection.

To meet these FOM a RaspberryPi was selected as the computing unit on-board the WAGN. The Pi has four USB ports on board as well as WiFi and a processor twice the strength of the Khepera IV, enough strength for an EKF and communication. There are also several General Purpose Input/Output (GPIO) pins including several for serial communication and others for external peripherals. The Pi can also run a more compact version of Ubuntu making the entire system more contiguous and easier to learn. However, the Pi does not have enough serial inputs to read all of the COZIR$^{\text{TM}}$ so another unit was needed to receive these inputs.

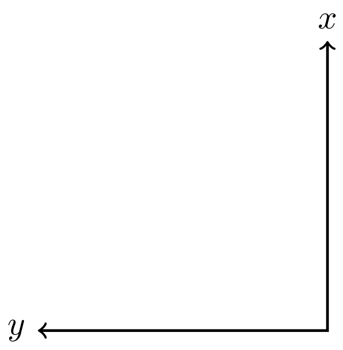
The only requirement for this unit is that it needed to have at least five serial ports (including one for communication with the RaspberryPi) and one analog port for the wind sensor. Arduino boards only have a maximum of three serial ports one was not used in the final design.[19] Instead a Teensy 3.5 was used because it has six available serial ports, a multitude of other inputs to handle other sensors like the anemometer, and it can be programmed like an Arduino making it easy to use and quick to set up.[20] All data collected by the Teensy was sent via USB communication to the RaspberryPi.

2.3 Extended Kalman Filter

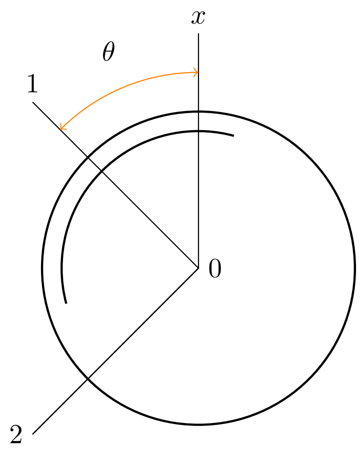
A Kalman Filter (KF) is a very useful tool to use for filtering out Gaussian noise of a system by using multiple sensors and fusing them and it is the optimal estimator for a linear dynamical system. However, when the dynamical system being estimated is non-linear, a normal KF no longer works. In this project, an Extended Kalman Filter is used for state estimation to account for both the encoder reading and IMU reading errors. The EKF works by linearizing the state in order to calculate the Kalman Gain via a normal KF. It then uses this Kalman Gain in a state estimation equation with the non-linear state to produce the updated state estimate.[21]

2.3.1 Linearization

The nonlinear state of the system in continuous time can be expressed by equation (1) and its discretized version is expressed in equation (2), where v_l and v_r are the velocities returned by the left and right encoder, respectively. In the equation, x and y are the the position coordinates of the robot in the reference frame of the CASE. Since this system will be analyzed as a discrete time system, the ΔT is the time between the current sensor readings and the previous sensor readings. Finally, θ is the angle between the direction the robot is traveling and the x-axis of the CASE. Figure 4 shows the reference frame of the robot with respect to the reference frame of the CASE.



(a) CASE.



(b) iRobot®.

Figure 4: Reference Frames.

$$\dot{X}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v(t) \cdot \cos(\theta(t)) \\ v(t) \cdot \sin(\theta(t)) \\ a(t) \cdot \cos(\theta(t)) \\ a(t) \cdot \sin(\theta(t)) \\ \omega(t) \end{bmatrix} \tag{1}$$

The above equation (eq. 1), in continuous time, can be converted to the following discrete time equation (eq. 2).

$$X(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \dot{x}(k+1) \\ \dot{y}(k+1) \\ \theta(k+1) \end{bmatrix} = f(X(k+1), u(k)) = \begin{bmatrix} x(k) + \frac{v_l+v_r}{2} \cdot \Delta T \cdot \cos(\theta(k) + \frac{v_l-v_r}{R} \cdot \Delta T) \\ y(k) + \frac{v_l+v_r}{2} \cdot \Delta T \cdot \sin(\theta(k) + \frac{v_l-v_r}{R} \cdot \Delta T) \\ \frac{v_l+v_r}{2} \cdot \cos(\theta(k) + \frac{v_l-v_r}{R} \cdot \Delta T) \\ \frac{v_l+v_r}{2} \cdot \sin(\theta(k) + \frac{v_l-v_r}{R} \cdot \Delta T) \\ \theta(k) + \frac{v_l-v_r}{R} \cdot \Delta T \end{bmatrix} \tag{2}$$

$$z(k+1) = h(X(k+1)) \tag{3}$$

In this system, equation (3) represents the Measurement Function and will be discussed later in this section. This state function can be simplified by expressing the control inputs as the controller u as shown in equation (4). This leads to the new state function, as shown in equation (5) by substituting the control values from equation (4) into equation (2).

$$u(k) = \begin{bmatrix} \Delta D(k) \\ V(k) \\ \Delta\theta(k) \end{bmatrix} = \begin{bmatrix} \frac{v_l+v_r}{2} \cdot \Delta T \\ \frac{v_l+v_r}{2} \\ \frac{v_l-v_r}{R} \cdot \Delta T \end{bmatrix} \tag{4}$$

$$f(X(k+1), u(k)) = \begin{bmatrix} x(k) + \Delta D(k) \cdot \cos(\theta(k) + \Delta\theta(k)) \\ y(k) + \Delta D(k) \cdot \sin(\theta(k) + \Delta\theta(k)) \\ V(k) \cdot \cos(\theta(k) + \Delta\theta(k)) \\ V(k) \cdot \sin(\theta(k) + \Delta\theta(k)) \\ \theta(k) + \Delta\theta(k) \end{bmatrix} \tag{5}$$

This non-linear state can then be linearized into the form of equation (6) through the use of the Jacobian matrix.

$$X(k+1) = \nabla f_x(k) \cdot X(k) + \nabla f_u(k) \cdot u(k) + n(k) \tag{6}$$

These Jacobians can be represented in a more traditional manner such that $\nabla f_x(k) = A$ and $\nabla f_u(k) = B$. By doing this, equation (6) can be written as equation (7).

$$X(k+1) = A \cdot X(k) + B \cdot u(k) + n(k) \tag{7}$$

In the above equations, $n(k)$ represents the Gaussian noise in the system.

In this linear dynamical system, the initial state estimate is defined by the measurements taken from the encoders, as shown above. In order to compare these measurements to another sensor, the accelerometer readings are taken and compared to a matrix called the "Expected Measurement Function".[21] This function calculates the expected acceleration readings based upon the encoders' readings. These expected values will be compared to the actual values returned by the accelerometer in the final state estimate equation (eq. (18)).

To calculate the value of the expected measurement function, first the appropriate matrix must be found to represent the accelerations in terms of the state, which contains distances and velocities. Normally, integration could be used to relate these values. However, since this system is a discrete time system, a technique referred to as pseudo-integration[8] must be used. This method asserts that if the time span ΔT is known and is very small (ie. $\Delta T \ll 1$ second), then the integral of a value v with respect to time, evaluated over that span of time, can be approximated as $\frac{v}{\Delta T}$.

As a result of this pseudo-integration method, the expected measurement function of the accelerometer can be expressed in terms of encoder values as shown in equation (8) (in the robot's frame of reference). In this equation, i_1 and j_2 represent the distance traveled along the 1 and 2 directions respectively, as shown in Figure 4. Similarly, X_{123} in this equation represents the state of the robot in the robot's frame of reference.

$$h(X_{123}(k+1)) = \begin{bmatrix} a_1 \\ a_2 \\ a_1 \\ a_2 \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta T^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\Delta T^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\Delta T^1} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\Delta T^1} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\Delta T^1} \end{bmatrix} \cdot \begin{bmatrix} i_1(k+1) \\ j_2(k+1) \\ \dot{i_1}(k+1) \\ \dot{j_2}(k+1) \\ \theta(k+1) \end{bmatrix} \tag{8}$$

This equation can then be changed to the CASE's frame of reference, which results in the expected measurement function shown in equation (9).

$$h(X(k+1)) = \begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \\ \omega \end{bmatrix} =$$

$$\begin{bmatrix} (\cos(\theta+\Delta\theta))^{-1} & (\sin(\theta+\Delta\theta))^{-1} & 0 & 0 & 0 \\ (\sin(\theta+\Delta\theta))^{-1} & (\cos(\theta+\Delta\theta))^{-1} & 0 & 0 & 0 \\ 0 & 0 & (\cos(\theta+\Delta\theta))^{-1} & (\sin(\theta+\Delta\theta))^{-1} & 0 \\ 0 & 0 & (\sin(\theta+\Delta\theta))^{-1} & (\cos(\theta+\Delta\theta))^{-1} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta T^{-2} \\ \Delta T^{-2} \\ \Delta T^{-1} \\ \Delta T^{-1} \\ \Delta T^{-1} \end{bmatrix} \cdot \begin{bmatrix} x(k+1) \\ y(k+1) \\ \dot{x}(k+1) \\ \dot{y}(k+1) \\ \theta(k+1) \end{bmatrix} \tag{9}$$

With the expected measurement function defined, the Jacobian of $h(X(k+1))$ can be found. This value can be used to express the measurement function in equation (10).

$$z(k+1) = \nabla h_x(X(k+1)) \cdot X(k+1) + w(k+1) \tag{10}$$

In order to convert this equation into a more traditional form, the Jacobian can be represented as $\nabla h_x(X(k+1)) = C$. This allows equation (10) to be rewritten as equation (11). In both of these equations, $w(k+1)$ represents the Gaussian noise of the sensor readings.

$$z(k+1) = C \cdot X(k+1) + w(k+1) \tag{11}$$

2.3.2 Predict

With the matrices calculated for both the linear and non-linear states, the prediction phase of the Extended Kalman Filter can begin. From this point on, the EKF will function much the same as a normal Kalman Filter, using the values from the linearized state in place of the non-linear state for calculating the Kalman Gain.

Firstly, the **Process Noise Covariance Matrix**, $Q(k)$, can be found by using the variance of the encoder ($\sigma^2_{encoder}$) with equation (12). This value can then be used to determine the **Process Error Covariance Matrix**, $P(k+1|k)$, as shown in equation (13). Equation (13) also requires the P-update value from the previous time step ($P(k|k)$).

$$Q(k) = B \cdot (\sigma^2_{encoder} \cdot I) \cdot B^T \tag{12}$$

$$P(k+1|k) = A \cdot P(k|k) \cdot A^T + Q(k) \tag{13}$$

In the code for the EKF, this prediction phase of the EKF is also used to define the predicted state, $\hat{X}(k+1|k)$, according to equation (14), which uses the previous $\hat{X}$-update ($\hat{X}(k|k)$) in the equation for the non-linear system.

$$\hat{X}(k+1|k) = f(\hat{X}(k|k), u(k)) \tag{14}$$

2.3.3 Update

The final stage of the EKF is to use the acquired information to update the state estimate. To start, the **Measurement Noise Covariance Matrix**, $R(k+1)$, must be calculated according to equation (15) in order to account for the accelerometer's accuracy. In equation (15), σ^2_{accel} is the variance of the accelerometer.

$$R(k+1) = \sigma^2_{accel} \cdot I \tag{15}$$

From here, the Kalman Gain can be found. Equation (16) shows the equation for the **Innovation Matrix**, $S(k+1)$, and equation (17) shows the equation for the **Kalman Gain**, $K(k+1)$, based on the inverse of the Innovation Matrix.

$$S(k+1) = C \cdot P(k+1|k) \cdot C^T + R(k+1) \tag{16}$$

$$K(k+1) = P(k+1|k) \cdot C^T S^{-1}(k+1) \tag{17}$$

With the Kalman Gain calculated, the update phase of the EKF can be completed by updating the State Estimate and the Process Error Covariance Matrix. The update of the State Estimate can be calculated in equation (18). This is the State Estimate which will be returned to the CASE by the EKF to inform it of where the robot is. It is also the State Estimate which will be passed on to the next iteration of the EKF. In this equation (eq. (18)), the $z(k+1)$ is the measurement of the accelerometer multiplied by a rotation matrix to bring it into the CASE's frame of reference. The update of the Process Error Covariance Matrix, calculated in equation (19), will also be passed on to the next EKF iteration. Once these values are calculated, this iteration of the EKF is complete.

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + K(k+1) \cdot (z(k+1) - C \cdot \hat{X}(k+1|k)) \tag{18}$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1) \cdot S(k+1) \cdot K^T(k+1) \tag{19}$$

2.4 Linear Kalman Filter

As mentioned in Section 2.3, the EKF is an excellent tool for state estimation of a nonlinear dynamical system. However, the EKF approximates the system as linear in order to calculate the Kalman Gain. As a result, the gain calculated is not the optimal gain for the system. Nevertheless, when presented with a nonlinear dynamical system, the EKF provides a state estimation which is accurate enough for navigation. For these reasons, the EKF was derived for this dynamical system by previous MQP teams[9], with reference to the University of New Mexico (UNM) report.[21]

However, it was recognized during this project that the only nonlinearities of the system arise as a result of the transformation from the robot's frame of reference to the CASE's frame of reference. Prior MQPs as well as the UNM report transform all values to the CASE's frame of reference from the beginning, resulting in the sines and cosines in the state matrix of equation (5). Yet this system can be made even simpler by moving the reference frame rotation step to *after* the Kalman Filter. By doing this, it allows the entire Kalman Filter to take place in the robot's frame of reference. By doing this, all of the rotational terms in equation (5) can be removed, leaving the system as shown in equation (20). This equation can be broken up into its state and control components as shown in equation (21).

$$f(X(k+1),u(k)) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \dot{x}(k+1) \\ \dot{y}(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k|k)+\Delta D(k) \\ y(k|k)+\Delta D(k) \\ V(k) \\ V(k) \\ \theta(k)+\Delta\theta(k) \end{bmatrix} \tag{20}$$

$$f(X(k+1),u(k)) = \begin{bmatrix} x(k|k) \\ y(k|k) \\ 0 \\ 0 \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \Delta D(k) \\ \Delta D(k) \\ V(k) \\ V(k) \\ \Delta\theta(k) \end{bmatrix} \tag{21}$$

In equation (20) and equation (21), the terms $\Delta D(k)$, $V(k)$ and $\Delta\theta(k)$ are the same as they are defined in equation (4). Additionally, the subscript 123 signifies that this is the state in the robot's reference frame, displayed as the 123 axes in Figure 4. As for the measurement function, the nonlinear transformation terms can also be removed from equation (9), which is just equation (8). Based on equations (21) and (8), the state in the robot's reference frame can be directly represented as shown in equations (22) and (23) after some simple matrix manipulation.

$$X_{123}(k+1|k) = A \cdot X(k|k) + B \cdot u(k) + n(k) \tag{22}$$

$$z(k+1) = C \cdot X_{123}(k+1) + w(k+1) \tag{23}$$

Where the values of A, B and C are consistent with equations (21) and (8) as shown below in equation (24).

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} \frac{1}{\Delta T^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\Delta T^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\Delta T^1} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\Delta T^1} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\Delta T^1} \end{bmatrix} \tag{24}$$

Just as in the EKF, this dynamical system is now represented in a linear form and can be passed through a Kalman Filter. Unlike the EKF, however, this linear system is a direct equality to the state transition function, not a linearization (ie. approximation). Since this is the case, that the dynamical system is linear in the robot's reference frame, it means that the Kalman Gain calculated through this method will be optimal for the system, unlike in the case of the EKF. A similar implementation of a Linear Kalman Filter can be found in the IEEE report[22] which will be discussed more in the Section 4. At this point, the equations for the Kalman Filter are almost identical to those presented in the EKF, so a brief summary will be provided below, with minor differences.

The prediction phase consists of the following equations:

$$\hat{X}_{123}(k+1|k) = A \cdot \hat{X}_{123}(k|k) + B \cdot u(k) \tag{25}$$

$$P(k+1|k) = A \cdot P(k|k) \cdot A^T + B \cdot (\sigma^2_{encoder} \cdot I) \cdot B^T \tag{26}$$

Following these steps, the update phase continues as follows:

$$S(k+1) = C \cdot P(k+1|k) \cdot C^T + \sigma^2_{accel} \cdot I \tag{27}$$

$$K(k+1) = P(k+1|k) \cdot C^T \cdot S^{-1} \tag{28}$$

$$\hat{X}_{123}(k+1|k+1) = \hat{X}_{123}(k+1|k) + K(k+1) \cdot (z(k+1) - C \cdot \hat{X}_{123}(k+1|k)) \tag{29}$$

$$P(k+1|k+1) = (I - K(k+1) \cdot C) \cdot P(k+1|k) \tag{30}$$

For a more in depth discussion on the equations of a Kalman Filter, see sections 2.3.2 and 2.3.3.

From equations (25) through (30), the value of $\hat{X}(k+1|k+1)$ will be the state estimate in the robot's frame of reference. Before transmitting this information to the CASE, it must first be transformed into something usable by the CASE. To do this, the state must be converted to the CASE's frame of reference. Since this system only has a rotation about the z axis, a simple Euler Angle rotation matrix about the 3-axis can be used for this rotation. Since there are two sets of 2-dimensional data being rotated (ie. x and y are being rotated as well as $\dot{x}$ and $\dot{y}$), a composite matrix consisting of multiple rotation matrices must be used. First, the rotation matrix about the 3-axis (going from 123 axes to XYZ axes) is shown in equation (31).

$$R_3 = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{31}$$

Next, the composite matrix can be constructed by transforming the group x and y as well as the group $\dot{x}$ and $\dot{y}$ while leaving θ unchanged, since it is independent of the reference frame.

$$\hat{X}_{XYZ} = \begin{bmatrix} \cos(\theta(k+1)) & \sin(\theta(k+1)) & 0 & 0 & 0 \\ -\sin(\theta(k+1)) & \cos(\theta(k+1)) & 0 & 0 & 0 \\ 0 & 0 & \cos(\theta(k+1)) & \sin(\theta(k+1)) & 0 \\ 0 & 0 & -\sin(\theta(k+1)) & \cos(\theta(k+1)) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \hat{X}_{123} \tag{32}$$

After passing the state through this rotation matrix, the value $\hat{X}_{XYZ}$ is the state estimate in the CASE's frame of reference. This value can then be passed off to the CASE in the same way that the state estimate was transmitted to the CASE after the EKF.

2.5 Plume Generation

For the WAGN to test the validity of the gas reconstruction algorithm, the experiment requires a system that can reliably produce a plume in the laboratory. The gas emission system must operate (1) safely, as exposure to high levels of CO_2 can be toxic,[9] (2) consistently, so that different experimental runs can be easily compared, and (3) in accordance with assumptions made by the plume reconstruction algorithm. The algorithm assumes the plume to be a point source in three dimensional space, so the diffuser in the system must simulate a single point of dispersion as closely as possible.

2.5.1 System Breakdown

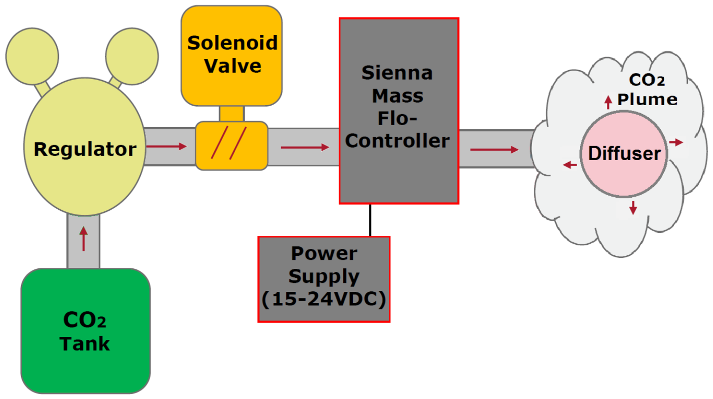
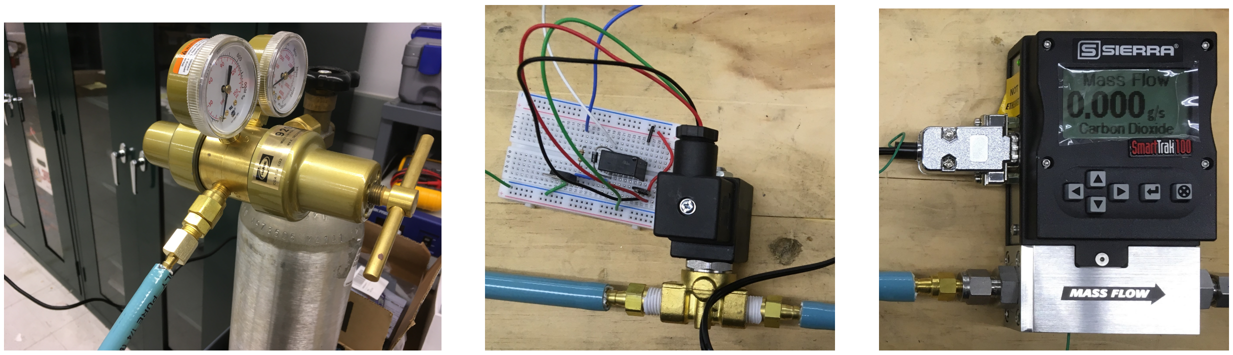


Figure 5: Plume Generation System Schematic.

The plume generation process begins with a tank of pressurized CO_2 at 700 psi, decreasing in pressure as gas is exhausted. In tandem with the tank, a Harris® Model 9296 Regulator is used to set a specific output pressure, so as the tank depressurizes the output pressure stays consistent.

After the regulator, the CO_2 moves through an Omega® SV3308 Solenoid Valve. The valve, when powered, is opened and closed using the push-button connected to the control circuit, closing in approximately. This provides a means to interrupt the flow that can be triggered much faster than closing the regulator or tank as a safety feature.



(a) Harris® Model 9296. (b) Omega® SV3300 and Control Circuit. (c) Sierra® SmartTrak.

Figure 6: Plume Generation Components.

Next the flow moves through a Sierra® SmartTrak C100L Mass Flow Controller. By measuring the change in temperature caused by gas moving past a heated wire, the controller gets a very accurate reading of the current mass flow, which is linked to an internal PID-controlled valve. Previous project teams have determined that the proper mass flow rate for this experiment is 500 $\frac{mg}{s}$[7, 9]. This yields a concentration that is still distinguishable from atmospheric levels at the detection ranges of the experiment, while not reaching toxic levels of CO_2 in the lab.

Finally, the flow moves through a diffuser as it is released out into the laboratory. The diffuser seeks to simulate a 3D point source as best as possible, which entails both omnidirectionality and negligible flow velocity. This is achieved using a wire mesh ball such that the flow can exit to the room in every direction. The flow velocity is affected by: (1) the output pressure of the regulator, (2) the friction losses as gas moves through the system, and (3) the surface area of the mesh ball. The pressure difference between the regulator and atmosphere is ultimately what is driving the flow, with friction from the gas interacting with the walls of the tubing helping to slow the gas. By entering into the chamber inside the diffuser ball, the flow is slowed significantly because the exit control surface is larger than its entrance. The ball is at the end of approximately two meters of flexible tubing, so it can be positioned away from the rest of the plume generation system, improving the quality of a point plume simulation.

2.5.2 Integration

Previous project teams planned to use LabVIEW® as a single interface to control all components of the gas emission system.[9] The intent was to consolidate user inputs onto the CASE to run alongside the WAGN control algorithm, creating a streamlined process for conducting the experiment. However, this concept was never fully implemented. The version of LabVIEW® for Linux does not support its Measurement and Automation Explorer^{TN} utility,[23] making sensor integration significantly more complicated than it would be on a computer running the Windows® version. Additionally, such an interface still would not offer complete control of the plume generation system because LabVIEW® cannot adjust the regulator. Instead, all components of the gas emission system are operated manually. This is not only the simplest option, but also allows for the most control.

2.6 Wind Tunnel

2.6.1 Concept

As was described in the Introduction, the end goal of this project is to recreate an outdoor environment in an experimental setting. Unlike a real-world scenario, the current plume generation system does not take into account wind speed or direction. In an effort to make a more accurate representation of this system, we conducted preliminary research towards creating a wind generation system with a test section large enough to encompass the WAGN as it performs the experiment.

The FOM driving the wind tunnel design are the size of the test section and wind speed which together determine the flow rate. A $2m$ x $2m$ x $6m$ test section was chosen because it provides enough space to conduct the experiment while minimizing fan size. A wind speed of up to $20\frac{m}{s}$ was chosen in order to simulate free stream flow in the atmosphere. In order to modeled the tunnel using Computational Fluid Dynamics (CFD), one must know boundary conditions for the flow. Upstream, this is the known pressure in the laboratory, $1atm$. Downstream, this is the volumetric flow rate required to achieve the desired test section velocity, $Q = 80\frac{m^3}{s}$.

A contraction upstream of the test section is a typical flow component in wind tunnels. It reduces variations in flow velocity and allows screens to be placed in a low speed region to reduce pressure losses. However, the space available in the room where the tunnel would reside does not support including a contraction. Contraction ratios normally range from 6 to 9 in small tunnels,[24] which would correspond to an inlet nearly $5m$ across. This is much higher than the room's ceiling. Even a more conservative contraction ratio would still require construction of a raised platform for the WAGN to operate on, rather than having it drive on the floor of the room. Because of the added complexity and limited benefit, the wind tunnel design should forgo a contraction.

2.6.2 Settling Chamber

The settling chamber is the region upstream of the test section that contains elements to attenuate turbulence and condition the flow. In our design, it shares the same cross section as the test section and contains both a honeycomb and a series of screens.

A grid with hexagonal cells (i.e. honeycomb) is used in the settling chamber to reduce swirl and lateral velocity in the flow. To determine cell size, there is a rule of thumb to fit $25,000$ total cells in the cross section.[24, 25] The cell hydraulic diameter (D_{cell}) can then be found from the cross section area (A_{cross}), cell area (A_{cell}), and cell side length (s_{cell}).

$$A_{cell} = \frac{A_{cross}}{25,000} \tag{33}$$

$$s_{cell} = 3^{1/4} \cdot sqrt(\frac{2 \cdot A_{cell}}{9}) \tag{34}$$

$$D_h = \frac{4 \cdot A_{cell}}{6 \cdot s_{cell}} \tag{35}$$

Two other design criteria have been found involving the ratio of cell axial length (L_h) to cell hydraulic diameter, and the grid's open area ratio. β_h[24, 26]

$$6 \leq \frac{L_h}{D_h} \leq 8 \tag{36}$$

$$\beta_h \geq 0.8 \tag{37}$$

Screens made of a fine wire mesh are used in the settling chamber to reduce variations in longitudinal flow velocity. They also break up turbulent eddies into smaller scale eddies that decay more rapidly. To avoid creating unstable flow, the screen's open area ratio (β_s) must be in the following range.[25]

$$0.58 \leq \beta_s \leq 0.8 \tag{38}$$

An individual screen's pressure drop coefficient (K_s) can then be found from the open area ratio and wire diameter (d), as well as the flow velocity (U) and fluid's kinematic viscosity. (ν)[24, 25] This results in a pressure loss proportional to the velocity head, a function of fluid velocity and density (ρ).

$$K_s = 6.5\,(\frac{1-\beta_s}{\beta_s^2})\,(\frac{Ud}{\beta_s \nu})^{-1/3} \tag{39}$$

$$\Delta P = \frac{1}{2}\rho U^2 \sum_n (K_s)_n \tag{40}$$

A screen's effect on flow quality is described by the turbulence reduction factor (f_s). This is not the absolute reduction in turbulence, but rather a comparison to how turbulence would be effected by the same flow section without any screens present. The factor's relation to the loss coefficient means it is much more efficient to use a series of screens with small losses rather than a single dense screen.[27] It has also been demonstrated that a series of screens should range from more coarse to more fine rather than using a single type of mesh.[26]

$$f_s = \sum_n \frac{1}{\sqrt{1 + (K_s)_n}} \tag{41}$$

Finally, the spacing between flow elements should be around 5% of the settling chamber diameter to let each screen act independently.[25]

2.6.3 Wind Tunnel Fan

In order to drive the large volume rate required for the tunnel, an axial fan from the Howden American Fan Company® should be considered. They offer fans up to $425,000CFM$,[28] which is well over the $170,000CFM$ required to achieve $20\frac{m}{s}$ in the test section.

3 Experiments and Results

3.1 Kalman Filter

Despite the Kalman filter returning somewhat accurate information, a substantial error arises as a result of the use of an IMU. This issue, noticed from the beginning of the tests, is that the state has a tendency to drift towards zero. Further investigation into this error revealed that the drift could be substantially slowed by increasing the value of σ_{accel}. In other words, the drift was minimized by minimizing trust in the IMU readings in the Kalman Filter. With this information, it was concluded that the drift is a result of the inaccuracies and drift of the IMU readings alone. As a result, pure odometry was used with the encoder readings for the navigation of the robot. There are two possible solutions to this problem, which will be discussed in Section 4.

3.2 Plume Analysis and Detection

The Plume Analysis and Detection (PAD) Instrument and Wireless, Autonomous Ground Navigator (WAGN) performed as planned, mapping out a path from the starting position to areas of high concentration in the experimental area. Figure 7 shows the plot generated by the Computer for Autonomous System Execution (CASE) from data received wirelessly from the WAGN during a plume experiment. The concentrations are mapped from 0 to 10000 Parts Per Million (PPM) on a linear color-based scale, green being 0 PPM to red being 10000 PPM, and the WAGN represented as a white circle.

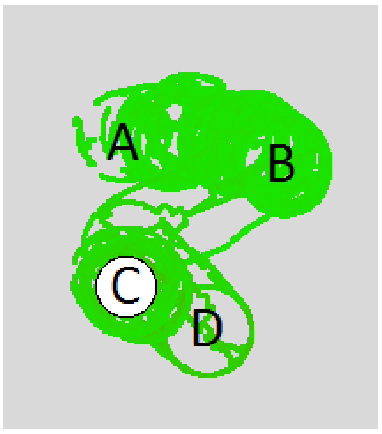


Figure 7: Low Strength Plume Experiment.

To test the accuracy and sensitivity of the PAD Instrument, a small, low concentration plume was introduced about a meter from the WAGN after a short period of time. The WAGN started at point A with no plume introduced and slowly moved to point B noticing a slightly higher gas concentration there. Then, when it had just about reached point B, a small plume was introduced into the experiment at point C. After a short delay, the robot moved to point C. The robot then momentarily moved to point D, likely because the plume wasn't fully defined, but then quickly returned to point C and circled around for the next few minutes, remaining there for the duration of the experiment.

3.3 Wind Tunnel Dimensions

Using the procedure outlined in Section 2.6.2, a settling chamber with the following specifications could meet design requirements. The honeycomb grid has a cell diameter $D_h = 1.36cm$ and cell length $L_h = 10cm$. Four screens following the honeycomb are described in Table 1. Each flow element is separated by $10cm$, giving the settling chamber a total length of $50cm$.

Table 1: Settling Chamber Screen Dimensions.

Screen	**1**	**2**	**3**	**4**	**total**
Open Area Ratio	0.6	0.6	0.6	0.6	-
Wire Diameter (mm)	0.85	0.53	0.36	0.25	-
Pressure Loss Coefficient	0.6	0.7	0.8	0.9	3.0
Turbulence Reduction Factor	0.79	0.77	0.75	0.73	0.33

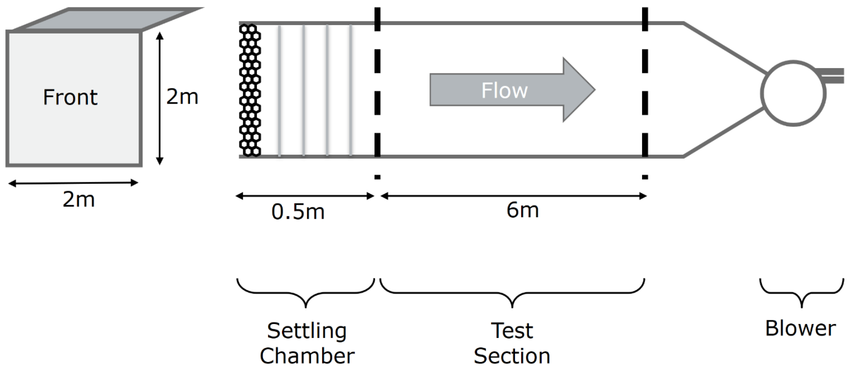


Figure 8: Wind Tunnel Design.

4 Conclusion

4.1 Plume Analysis and Detection

Upon experimenting with the Wireless, Autonomous Ground Navigator (WAGN) and Kalman Filter (KF) it was found that the Inertial Measurement Unit (IMU) introduced a significant amount of noise into the system even with the filter. As a result, the plume recreation experiment was performed solely using the encoders for localization and was more accurate in routine use than the sensor fusion model. Despite that, the model generated by the Computer for Autonomous System Execution (CASE) simulated the environment correctly and followed closely to the intended design. Although the plume introduced only had a small concentration increase over ambient CO_2, the robot navigated accurately and generated a correct model in Figure 7. However, improvements could be made to the algorithm to provide a more robust search.

Future work on the navigators includes implementing Simultaneous Localization and Mapping (SLAM), a multi-robot system, or a Wireless, Autonomous Aerial Navigator (WAAN). SLAM would allow the WAGN to actively avoid objects in its path and could navigate a non-linear path to the plume and could make use of the existing built in localization. This would allow for more a more dynamic experiment, more similar to a real-world scenario. Implementing a multi-robot system would be easily scalable from the current framework implementing an iRobot® Create 2 to a system of several and would allow for a quicker, more robust Plume Analysis and Detection (PAD). A WAAN could help bring the project into its original scope and would allow for a switch from a two dimensional to a three dimensional model which would contain much more valuable data for analyzing the impact of a plume on the environment.

4.2 System Filtering

As discussed thus far, both the EKF and the LKF do not increase the accuracy of the navigational system while using the IMU. This comes as a result of the high levels of noise and inaccuracy of the accelerometer. However, in the IEEE report titled ”Localization of iRobot Create using inertial measuring unit”,[22] a different type of filter is suggested called a Complementary Filter. It is stated in the paper that the Complementary filter has several advantages over the Kalman Filter when working with the high noise of an accelerometer. This type of filter was not used in this report because the source was not discovered until late into the MQP. However, it is recommended that, if future MQPs would like to continue using the IMU for navigation, that they investigate further into this ”Complementary Filter”

4.3 Alternative Sensors

A second approach to avoiding the inaccuracies encountered with the current system is to avoid the use of an IMU. Of all the sensors which could be used for navigation, an accelerometer is by far one of the noisiest sensors. By replacing the IMU with a less noisy sensor, such as IR sensors mounted around the chassis to detect walls or a LIDAR system, the Kalman Filters derived in this paper could still be used for state estimation by simply changing the Measurement Function.

Additionally, to successfully operate in a windy environment, the WAGN requires an anemometer other than the Rev.P. This is because its accuracy has a strong dependence on wind direction, and it only functions well within a narrow range.[29] Future projects should strongly consider sonic anemometers such as the DS-2 or FT702LM, since they offer much greater stability over a full range of wind directions.[6]

4.4 Wind Tunnel Simulation

The next step in the wind tunnel design process would be to conduct a CFD simulation which follows the boundary conditions outlined in Section 2.6.1. Using these constraints, as well as modeling the honeycomb grid and including the pressure drops from each screen found in Section 3.3, one can obtain a much better understanding of the boundary layer size in the wind tunnel to see if it is suitable for the experiment or requires modification.

References

[1] Michael A Demetriou, Nikolaos A Gatsonis, and Jeffrey R Court. Coupled controls-computational fluids approach for the estimation of the concentration from a moving gaseous source in a 2-d domain with a lyapunov-guided sensing aerial vehicle. *IEEE Transactions on Control Systems Technology*, 22(3):853–867, 2014.

[2] Dominique Martinez and Laurent Perrinet. Cooperation between vision and olfaction in a koala robot. In *Report on the 2002 Workshop on Neuromorphic Engineering*, pages 51–53, 2002.

[3] Takamichi Nakamoto, Hiroshi Ishida, and Toyosaka Moriizumi. Peer reviewed: A sensing system for odor plumes. *Analytical chemistry*, 71(15):531A–537A, 1999.

[4] Emmanuel A Gonzalez, Floyd Matthew G Mascenon, Alexis J Magpantay, Kervin Clyde C Go, and Miguel Lorenzo J Cordero. Design of an autonomous mobile olfactory robot for chemical source localization. In *TENCON 2004. 2004 IEEE Region 10 Conference*, volume 500, pages 475–478. IEEE, 2004.

[5] Tatiana Egorova, Nikolaos A Gatsonis, and Michael A Demetriou. Estimation of gaseous plume concentration with an unmanned aerial vehicle. *Journal of Guidance, Control, and Dynamics*, pages 1314–1324, 2016.

[6] MAD1501, Mica M. Anglin, Mitchell A. Hunt, and Matthew T. Myles. Gas source localization with a mobile sensing ground vehicle, 2015.

[7] NAG1501, Christopher G. Clark, Mitchell T. Greene, and Madeline D. Seigle. Design of a plume generation and detection systems, 2015.

[8] MAD1601, Nicholas C Christie, and John W Colfe. Plume detection using a gas-sensing mobile robot, 2016.

[9] NAG1602, Michael Barney, and Stephanie Rivard. Design and integration of an indoor plume experimental setup, 2016.

[10] Ubuntu. `https://www.ubuntu.com`. Web. 19 Feb 2017.

[11] Python. `https://www.python.org`. Web. 19 Feb 2017.

[12] Bluetooth vs Wifi. `http://www.diffen.com/difference/Bluetooth_vs_Wifi`. Web. 19 Feb 2017.

[13] TCP vs UDP. `http://www.diffen.com/difference/TCP_vs_UDP`. Web. 19 Feb 2017.

[14] Edwin Olson. AprilTags Visual Fiducial System. The APRIL Robotics Laboratory at the University of Michigan. `https://april.eecs.umich.edu/software/apriltag.html`. Web. 19 Feb 2017.

[15] OpenCV. http://opencv.org. Web. 19 Feb 2017.

[16] K-Team Khepera IV. `https://www.k-team.com/mobile-robotics-products/khepera-iv`. Web. 19 Feb 2017.

[17] iRobot®. *iRobot Create® 2 Open Interface Specification based on the iRobot Roomba® 600.* Web. 19 Feb 2017.

[18] *COZIR^{TM} Software User's Guide.* Web. 19 Feb 2017.

[19] Arduino. `https://www.arduino.cc/`. Web. 19 Feb 2017.

[20] Teensy. `https://www.pjrc.com/teensy/`. Web. 19 Feb 2017.

[21] Thomas J. Otahal and Herbert G. Tanner. An extended kalman filter implementation for the khepera ii mobile robot. Technical report, UNM, 2008.

[22] Matej Guráň, Tomáš Fico, Anežka Chovancová, František Duchoň, Peter Hubinský, and Dúbravský Dúbravský. Localization of irobot create using inertial measuring unit. Technical report, IEEE, 2014.

[23] NI-VISA Support in Linux. `http://digital.ni.com/public.nsf/allkb/EBEAFAC0AED349FE862573A5007731E2`. Web. 16 Mar 2017.

[24] Rabindra D Mehta and Peter Bradshaw. Design rules for small low speed wind tunnels. *The Aeronautical Journal (1968)*, 83(827):443–453, 1979.

[25] James H Bell and Rabindra D Mehta. Design and calibration of the mixing layer and wind tunnel. Technical report, NASA, 1989.

[26] S Brusca, R Lanzafame, and M Messina. Low-speed wind tunnel: design and build. *Wind Tunnels: Aerodynamics, Models and Experiments*, pages 189–220, 2011.

[27] Hugh L Dryden and Ira H Abbott. The design of low-turbulence wind tunnels. NACA TN 1755, Nov. 1948.

[28] JM - Tubeaxial Fans. `http://www.americanfan.com/fans/axial-fans/tubeaxial-fans.html`. Web. 16 Mar 2017.

[29] D Prohasky and S Watkins. Low cost hot-element anemometry verses the tfi cobra. In *19th Australasian Fluid Mechanics Conference*, 2014.

Appendices

A Nomenclature

J	Jacobian Matrix
∇M	Jacobian of Matrix M
$\hat{v}$	Estimate of Value v
Δx	Variable displacement vector
α	Acceleration, m/s^2

Subscript

x	Jacobian with respect to the state vector, X
u	Jacobian with respect to the control vector, U
k	Current Time Step in Kalman Filter
$(k\|k)$	Previous Update in Kalman Filter
$(k+1\|k)$	Current Prediction in Kalman Filter
$(k+1\|k+1)$	Current Update in Kalman Filter

B Acronyms

ABS Acrylonitrile butadiene styrene

CASE Computer for Autonomous System Execution

CFD Computational Fluid Dynamics

CG Center of Gravity

DOF Degree of Freedom

EKF Extended Kalman Filter

FOM Figures of Merit

GPIO General Purpose Input/Output

IMU Inertial Measurement Unit

IR Infrared

KF Kalman Filter

PAD Plume Analysis and Detection

PPM Parts Per Million

SLAM Simultaneous Localization and Mapping

TCP Transmission Control Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

WAAN Wireless, Autonomous Aerial Navigator

WAGN Wireless, Autonomous Ground Navigator

WLAN Wireless Local Area Network

WPI Worcester Polytechnic Institute

C Code

For all the code associated with this projects, as well as instructions on how to run it, please visit `https://github.com/rmwiesenberg/MQP`. The repository has the code from past projects, all the steps of this one (iRobot Creates 1 and 2), and the beginnings of the next steps for the project. Please refer to the READMEs for run instructions, the code is detailed in the files themselves.