Modeling, Simulation, and Design of a Time-Delayed Multi-Agent System of Vehicles with Time Delays

by

NICHOLAS GAMEZ, M.S.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

COMMITTEE MEMBERS:
Mo Jamshidi, Ph.D., Chair
Ahmad Taha, Ph.D.
Jeff Prevost, Ph.D.
Paul Rad, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Engineering
Department of Electrical and Computer Engineering
May 2017
DEDICATION

I would like to dedicate this thesis to my family for all the support throughout my studies.
Modeling, Simulation, and Design of a Time-Delayed Multi-Agent System of Vehicles with Time Delays

Nicholas Gamez, B.A.
The University of Texas at San Antonio, 2017

Supervising Professor: Mo Jamshidi, Ph.D.

With the advent of cheap wireless communication and advancement in cloud computing, controllers are continually being placed off the systems they stabilize. A physical distance between the controller and plant brings in a new issue for modeling and designing control environments. Time delays slow the exchange of information to non-negligible values and must be accounted for when implementing new control systems. When considering mobile robotic platforms, a delay can lead to the mobile agents falling off cliffs, driving in restricted areas, or lose flight stability. A solution to account for the time delays will be presented by accounting for a set update interval before the controller is updated in a discrete fashion.

Presented in this thesis is a thorough look at a single unmanned ground vehicle and the maximum bound on the update time of a controller before the system becomes unstable. The ground vehicle is controlled first by a state feedback controller and then with observer based feedback and each maximum bound is found and simulated. A new environment is set up to include a flying robot to send the ground vehicle points of interest through a wireless network to test the validity of the controller given another vehicle increasing the burden on the wireless network.
# TABLE OF CONTENTS

**Acknowledgements** .......................................................... iv

**Abstract** .............................................................................. v

**List of Tables** ................................................................. viii

**List of Figures** ................................................................. ix

**Chapter 1: Introduction** ...................................................... 1

**Chapter 2: Network Control Systems and Stability** .............. 2

  2.1 Current Network Control Research .................................. 2

  2.1.1 Maximum Allowable Time Interval ................................ 3

  2.1.2 Improved MATI equation ............................................ 4

  2.1.3 Model Based Network Control ..................................... 5

  2.1.4 Unknown Input Observer for Networked Control Systems 5

  2.2 Chosen Approach .......................................................... 6

  2.2.1 Full State Feedback .................................................. 6

  2.2.2 Output Feedback ...................................................... 8

  2.2.3 State Feedback with Network Delay ............................ 9

**Chapter 3: Modeling of the Mobile Robotic Agents** .............. 12

  3.1 Unmanned Ground Vehicle - Differential Drive ............... 12

  3.1.1 Mathematical Modeling ............................................ 12

  3.1.2 Control ............................................................... 14

  3.1.3 Three Part Controller .............................................. 14

  3.1.4 Polar Transformation Controller ................................ 17

  3.2 Unmanned Aerial Vehicle - Quad-Copter ......................... 20
LIST OF TABLES

Table 1.1 Benefits and Drawbacks of Networked Control Systems . . . . . . . . . . . . . . . 1
LIST OF FIGURES

Figure 2.1 Model of a Networked Control System . . . . . . . . . . . . . . . . . . . . 2
Figure 2.2 NCS with Plant Dynamics Modeled in the Controller . . . . . . . . . . . . 7
Figure 2.3 NCS with Plant Dynamics Modeled in the Controller . . . . . . . . . . . . 8
Figure 2.4 A networked control system with communication delays present . . . . . . 10

Figure 3.1 Differential Drive UGV . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
Figure 3.2 Response of the UGV with the three part controller . . . . . . . . . . . . . 16
Figure 3.3 Inputs sent to UGV from the three part controller . . . . . . . . . . . . . . 16
Figure 3.4 Outputs of the UGV with the single polar controller . . . . . . . . . . . . 18
Figure 3.5 Inputs sent to the plant by the polar controller . . . . . . . . . . . . . . . 18
Figure 3.6 UGV with observer based feedback and the error associated with each state 19
Figure 3.7 Inputs required to drive the observer based UGV . . . . . . . . . . . . . . 20
Figure 3.8 Quadrotor representation . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
Figure 3.9 Response of a quadcopter UAV with an LQR state feedback controller . . 24
Figure 3.10 3D response of the quadcopter . . . . . . . . . . . . . . . . . . . . . . . . 24
Figure 3.11 Inputs required for control of UAV to a hovering position . . . . . . . . . 25

Figure 4.1 Max eigenvalues for a UGV relative to the update time of the controller . . 28
Figure 4.2 State response for UGV with state feedback control and $h = 0.1 \text{ s}$ . . . 29
Figure 4.3 Inputs for UGV with state feedback control and $h = 0.1 \text{ s}$ . . . . . . . . . 29
Figure 4.4 State response for UGV with state feedback control and $h = 1.0 \text{ s}$ . . . 30
Figure 4.5 Inputs for UGV with state feedback control and $h = 1.0 \text{ s}$ . . . . . . . . . 30
Figure 4.6 State response for UGV with state feedback control and $h = 2.0 \text{ s}$ . . . 31
Figure 4.7 Inputs for UGV with state feedback control and $h = 2.0 \text{ s}$ . . . . . . . . . 31
Figure 4.8 Max eigenvalues for a UGV relative to the update time of the controller . . 32
Figure 4.9 Observer based system response with update time of 0.1 seconds . . . . . . 33
Chapter 1: INTRODUCTION

Recently, the blossoming of the Internet of things (IoT) has created a bevy of connected nodes within a system sharing information through wireless communication protocols \[1\]. Within these highly interconnected systems, the time delay due to the network is not small enough to ignore.

Networked control systems are becoming more common with an increased reliance on the Internet, especially with cloud computing becoming consensus favorite for large computations without physical hardware demands. Robotics has begun to offload computations to the cloud creating smarter robots without larger hardware demands \[2\]. A networked control system has various benefits and drawbacks described in Table 1.1.

<table>
<thead>
<tr>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced Wiring</td>
<td>Non-synchronous Control</td>
</tr>
<tr>
<td>Easy Maintenance</td>
<td>Packet Loss over Network</td>
</tr>
<tr>
<td>Increased System Agility</td>
<td>May split data into multiple packets</td>
</tr>
</tbody>
</table>

The research presented details the characterization of an update time for a mobile robotic controller which will allow for delays to propagate without affecting the controller performance. A system of two robotic agents is then simulated to mimic a search and rescue task with one agent sending a destination to the other agent over a wireless network. The study compares the effect of compensating for the expected delay against no compensation. This highlights the benefits of compensating for a time delay with a mobile robotic platform not used before.

The rest of the thesis is organized as follows. Chapter 2 explains the literature review over networked control systems with an emphasis towards the maximum time interval a system can endure with stable results. Chapter 3 defines the robotic agents chosen and the modeling and control required for each agent. Chapter 4 follows with the simulation results for the networked control values and the multi agent system. Lastly, chapter 5 details the conclusion and future work associated with the advancement of time delay theory with respect to mobile robotics.
Chapter 2: NETWORK CONTROL SYSTEMS AND STABILITY

2.1 Current Network Control Research

Classical control systems are modeled with the assumption that all information is available to a plants, actuators, and controllers instantaneously. In real time systems, the computation and communication time can degrade the system performance and even lead to instability. For a networked system the introduction of communication delays further degrades the assumption of instantaneous dissemination of information. Introducing a network into a collection of actuators, sensors, and controllers brings about delays changing the dynamic model of the system.

Figure 2.1 shows a general system setup with a network between the plant and controller. There are two time delays associated with this networked control system, the communication time from sending the sensor information to the controller and the time from controller to actuator denoted $\tau_{sc}$ and $\tau_{ca}$ respectively. During modelling, the computational time requirement can be lumped together with either of the communication delays.

![Figure 2.1: Model of Networked Control System with network delays between the plant and controller.](image)

There are two main approaches when considering the control of time delayed systems.

1. Create the control law without considering the communication issues and design the com-
communication that minimizes the chance of data loss

2. Treat the network as a given condition and design the control with the conditions in mind

For instance, one can design a system which requires control updates intermittently where the update time is larger than the highest expected value of network delay. The following sections outline the different techniques to characterize an update interval and modeling the system to create a stable closed loop system. The following sections shows some of the recent studies conducted to account for the network in the loop systems that are becoming commonplace in robotics and controls.

2.1.1 Maximum Allowable Time Interval

Networked control systems do not have a set time delay for any given communication protocol. The time delay is always varying dependent on the communication protocol, network load, and bandwidth available. One method for combating the varying delays is to find an upper bound on the update time the system can tolerate before there are undesirable results. An approach for this upper bound, $\tau$, to preserve stability of a closed loop system was defined in [3] as the maximum allowable transfer interval, MATI.

$$
\tau < \frac{\lambda_{\min}(Q)}{16\lambda_{\max}(P)\sqrt{\lambda_{\max}(P)/\lambda_{\min}(P)}||A||^2(\sqrt{\lambda_{\max}(P)/\lambda_{\min}(P)} + 1) \sum_{i=1}^{p} i}
$$

(2.1)

In equation (2.1), $p$ is the number of nodes within the system requiring communication through the network. $P$ and $Q$ are found through the Lyapunov Equation $A_{11}^TP + PA_{11} = -Q$, where the system is defined as

$$
\dot{z}(t) = \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
x(t) \\
e(t)
\end{bmatrix}
$$

(2.2)

where the system states are comprised of the plant and controller states, $x(t) = [x_p(t)x_c(t)]$, while the error states, $e(t)$, are derived by the error from the last updated network values. The state space
representations for the plant and controller are shown in equation 2.3.

\[
\begin{align*}
\dot{x}_p &= A_p x_p + B_p u \\
\dot{x}_c &= A_c x_c + B_c y + B_c e_y \\
y &= C_p x_p \\
u &= C_c x_c + D_c y + D_c e_y \\
\end{align*}
\]

(2.3)

Using the state spaces from equation 2.3, the values of the A matrix in equation 2.2 are shown below [4].

\[
\begin{align*}
A_{11} &= \begin{bmatrix} A_p + B_p D_c C_p & B_p C_c \\ B_c C_p & A_c \end{bmatrix} \\
A_{12} &= \begin{bmatrix} B_p D_c \\ B_c \end{bmatrix} \\
A_{21} &= [-C_p \ 0] A_{11} \\
A_{22} &= [-C_p \ 0] A_{12} \\
\end{align*}
\]

(2.4)

Generally, the MATI value is very conservative with systems able to tolerate orders of magnitude higher update intervals [5].

### 2.1.2 Improved MATI equation

Nesic and Teel [6] improve upon the MATI bound by stating the input-output \( L^\infty \) stability of a system is preserved if

\[
\tau < \frac{1}{L} \ln \left( \frac{L + \gamma}{\rho L + \gamma} \right).
\]

The communication protocol is quantified by \( \rho \), and \( L \) is the growth of the network error between real inputs and outputs and the network values. The \( L^\infty \) disturbance gain capturing the system robustness without a network is described by \( \gamma \). The MATI values give a guaranteed stable upper bound for a system. In simulation, the system may still tolerate higher update times, but the settling time and/or the input values could be undesirable for the system. Specifically for the issue of trajectory planning in mobile robotics, a robot cannot realistically change the velocity of their wheels by 10 m/s where a higher update time may demand such movements.
2.1.3 Model Based Network Control

Model based networked control systems are designed to approximate the dynamics of the plant at the controller instead of sending the dynamics over the network. This, in turn, reduces the network load and allows the system to tolerate higher update times [7]. In this model, the system does not consider delay between the controller and actuator. This case occurs when the actuator and controller are located at the same position or are computed on the same device.

The system is predicated on the update time, $h$, when the sensor data is sent to the controller over the network. Lowering the update time reduces the bandwidth needed in the network and frees the network to allocate resources elsewhere. For example, in a large scale system with multiple vehicles with a centralized controller, each vehicle needs to correspond with the controller stressing the network. By limiting the amount of updates of each vehicle, the network can handle an increased number of vehicles before the degradation of the system becomes unmanageable. Compared to the MATI bound, the maximum bound for the update time, $h$, is 3-5 orders of magnitude larger.

2.1.4 Unknown Input Observer for Networked Control Systems

Taha et. al. in [8] studied an unknown input observer (UIO) and derived the dynamics for a networked UIO in order to minimize the effect of unknown inputs or disturbances to the system. Furthermore, a maximum time-delay bound was found much like the MATI bounds explained above. The research focuses on the characterization of delays as perturbation to the system and finding the bounds of this perturbation.

For example, given a closed loop networked system given by equation,

$$
\dot{x} = \Lambda_r \ast x + \delta \Lambda_r + \Omega u_2
$$

(2.5)

where $\Lambda_r$ is the closed loop state matrix, $\delta \Lambda_r$ is the perturbation due to the network, and $\Omega$ is the unknown input perturbation. The bound on the perturbation was determined by the Lyapunov
equation with matrices $P = P^T > 0$ and $Q = Q^T > 0$.

$$\Lambda_r^T P + P \Lambda_r = -2Q$$

If $||u_x|| < \mu||x||$ where $\mu > 0$ and $\lambda_{min}(Q) - \mu \lambda_{max}(P)||\Omega|| > 0$ then if the norm of the perturbation satisfies the inequality

$$||\delta \Lambda_r|| = \frac{\lambda_{min}(Q) - \mu \lambda_{max}(P)||\Omega||}{\lambda_{max}(P)}$$

the system is globally exponentially stable around the origin.

### 2.2 Chosen Approach

After the literature review over the current mathematical representations of networked time delay systems, the model based NCS approach was chosen to be implemented for the mobile robots. The concept of maximum allowable time interval, MATI, was also chosen to find a baseline delay to base the robustness of the control system. Since the MATI value is not the absolute highest update time, it will just be used to show the stability of the next control algorithms for the most conservative theoretical values before finding an actual maximum time interval. The next sections expand on these model based network control theory.

#### 2.2.1 Full State Feedback

Model based control of a state feedback system is the easiest algorithm to define. Here all the states are available to the controller and any delay is negligible. The controller will update the control inputs every $h$ seconds. The control is considered closed loop during the updating of the controller and open loop with the modeled dynamics in the controller driving the states while waiting to be updated again. The system shown in Figure [2.2] has an error in the states defined by $e = x - \hat{x}$ and the matrix errors as $\tilde{A} = A - \hat{A}$, $\tilde{B} = B - \hat{B}$. The controller is updated at every time $t_k$ where $t_{k+1} = t_k + h$. With the updating of the model at every $t_k$ seconds, the error states are set to 0 at
every update.

\[ \begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A + BK & -BK \\ \bar{A} + B\bar{K} & \bar{A} - B\bar{K} \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} + \begin{bmatrix} x(t_k) \\ e(t_k) \end{bmatrix} = \begin{bmatrix} x(t_k) \\ 0 \end{bmatrix} \quad \forall t \in [t_k, t_{k+1}] \quad (2.6) \]

which can be rewritten as \( \dot{z} = \Lambda z \) for \( t \in [t_k, t_{k+q}) \).

Montestruque and Antsaklis [7] state two theorems regarding the model based NCS with full state feedback.

1. For a system with initial conditions \( z(t_0) = \begin{bmatrix} x(t_0) \\ 0 \end{bmatrix} \) and defined by equation (2.6), the response becomes

\[
\begin{align*}
  z(t) &= e^{\Lambda(t-t_k)} M^k z(t_0) \\
  t &\in [t_k, t_{k+1}) \\
  t_{k+1} &= h + t_k \\
  M &= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} e^{\Lambda h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}
\end{align*}
\quad (2.7)
\]
2. The system in equation is globally exponentially stable around $z = \begin{bmatrix} x \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ if the eigenvalues of $M$ are inside the unit circle.

The second theorem is useful for finding the highest update time, $h$, the system can tolerate before the output becomes undesirable. For the mobile robots trajectory tracking case, defining the system in this way is very useful to drive the states to 0. The mobile robot can be designed such that the final destination is located at the origin and the system must be driven to 0.

2.2.2 Output Feedback

The same research characterizes the stability and response of a system without all the states available to the controller. A Luenberger observer can be used to estimate the plant states not available. An observer replaces the sensor in the system shown in Figure 2.2. Figure 2.3 shows the updated system with the Luenberger observer placed within the sensor.

![Figure 2.3](image)

*Figure 2.3:* Diagram for a model based network control system where the plant dynamics are modeled in the controller.

The error vector now becomes the error between the model and the observer, $e = \bar{x} - \hat{x}$. 

8
Similarly to before, the system is \( \dot{z} = \Lambda z \) where,

\[
z = \begin{bmatrix} x \\ \bar{x} \\ e \end{bmatrix}, \quad \Lambda = \begin{bmatrix} A & BK & -BK \\ LC & \hat{A} - L\hat{C} + \hat{B}K + L\hat{D}K & -\hat{B}K - L\hat{D}K \\ LC & L\hat{D}K - L\hat{C} & \hat{A} - L\hat{D}K \end{bmatrix}, \quad t \in [t_k, t_{k+1})
\]

This model based approach is advantageous in the stability analysis and responses of each approach are all similar in nature. For the observer output based feedback, the system is stable if all the eigenvalues of the matrix \( M \) are within the unit circle where,

\[
M = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} e^{Ah} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

The response is then characterized in equation (2.9)

\[
z(t) = e^{A(t-t_k)} M^k z_0, \quad t \in [t_k, t_{k+1}) \tag{2.9}
\]

### 2.2.3 State Feedback with Network Delay

The next extension for model based NCS is to consider the network delays when transmitting messages in a congested or slower medium. A congested network will cause the data to be held until the previous information is sent. A slower network can be caused by a lack of bandwidth or the size of the data being sent. In this case, the update time is still labeled \( h \) and a new variable, \( \tau \), denotes the network time. Now the networked control system is modeled as shown in (2.4) \( \ddot{x} \)
The assumption that the time delay is less than the update time lessens computational complexity of the system allowing the system to be described below:

\[
\begin{align*}
\text{Plant} : & \quad \dot{x} = Ax + Bu \\
\text{Model} : & \quad \dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u \\
\text{Controller} : & \quad u = K\hat{x} \quad t \in [t_k, t_{k+1}) \\
\text{Propagation} : & \quad \ddot{x} = \hat{A}\dot{x} + \hat{B}u \quad t \in [t_{k+1} - \tau, t_{k+1}]
\end{align*}
\quad (2.10)
\]

The propagation unit is updated with the network delayed plant states at \(t_{k+1} - \tau\), and the propagation states are sent to the model at \(t_{k+1}\). There are now two errors to keep track of, \(\hat{e} = \ddot{x} - \hat{x}\) and \(\hat{e} = x - \ddot{x}\) so the system \(\dot{z} = \Lambda z\) is defined by:

\[
z = \begin{bmatrix} x \\ \hat{e} \\ \hat{\hat{e}} \end{bmatrix}
\]
\[
\Lambda = \begin{bmatrix}
A + BK & -BK & -BK \\
(A - \hat{A}) + (B - \hat{B})K & \hat{A} - (B - \hat{B})K & -(B - \hat{B})K \\
0 & 0 & \hat{A}
\end{bmatrix}
\]

with dynamics described below.

\[
z(t_k) = \begin{bmatrix}
x(t_k) \\
\dot{e}(t_k) \\
0
\end{bmatrix} 
\quad t \in [t_k, t_{k+1} - \tau)
\]

\[
z(t_{k+1} - \tau) = \begin{bmatrix}
x(t_{k+1} - \tau) \\
0 \\
\dot{e}(t_{k+1} - \tau) + \dot{e}(t_{k+1} - \tau)
\end{bmatrix} 
\quad t \in [t_{k+1} - \tau, t_{k+1})
\] (2.11)

For the newly defined system with networked time delays considered has the response:

\[
z(t) = e^{\Lambda(t-t_k)}M^kz(t_0) 
\quad t \in [t_k, t_{k+1} - \tau)
\]

\[
z(t) = e^{\Lambda(t-t_{k+1}+\tau)} \begin{bmatrix}
I & 0 & 0 \\
0 & 0 & 0 \\
0 & I & I
\end{bmatrix} e^{\Lambda(h-\tau)}M^kz(t_0) 
\quad t \in [t_{k+1} - \tau, t_{k+1})
\] (2.12)

\[
M = \begin{bmatrix}
I & 0 & 0 \\
0 & I & 0 \\
0 & 0 & 0
\end{bmatrix} e^{\Lambda\tau} \begin{bmatrix}
I & 0 & 0 \\
0 & I & 0 \\
0 & 0 & I
\end{bmatrix} e^{\Lambda(h-\tau)}
\]

Much like the state feedback case above, if \( M \) has eigenvalues within the unit circle, the system will be globally exponentially stable. This system definition will be applied to the mobile system to determine the maximum value of \( \tau \) given different update times, \( h \), that the UGV can handle before it will not reach its destination. This will be compared to the case where no compensation is given to the time delays to show the increased tolerance of time delays with the aforementioned modeled network control system theory.
Chapter 3: MODELING OF THE MOBILE ROBOTIC AGENTS

Before the communication dynamics are considered, the mobile robotic subsystems must be modeled. The two separate mobile agents are mathematically developed in this chapter. The unmanned ground vehicle (UGV) is first defined and transformed into a polar coordinate system followed by three part state feedback controller. Next, the unmanned aerial vehicle (UAV), more specifically a quad-copter, is modeled, linearized, and controlled to hover with a LQR state feedback controller.

3.1 Unmanned Ground Vehicle - Differential Drive

3.1.1 Mathematical Modeling

Many UGV’s are created utilizing the differential drive mechanism. Two wheels, each moving independently, can create three different driving motions. Given Figure 3.1[9], the three motions are given below:

1. Linear movement in a straight line when both wheels have equal velocity \(V_r = V_l\)

2. Radial motion when both wheels have equal magnitude but opposite direction \(V_r = -V_l\)

3. When the wheels are not equal velocity \(V_r \neq V_l\) the robot will move in a circular motion around some radius of curvature

The kinematic model of the differential drive with respect to Earth’s coordinate system can be described according to Cook’s description in [10].

\[
\begin{align*}
\dot{x} &= -\frac{v_r + v_l}{2} \sin \theta \\
\dot{y} &= \frac{v_r + v_l}{2} \cos \theta \\
\dot{\theta} &= \frac{v_r - v_l}{l}
\end{align*}
\] (3.1)
The system is defined by two control variables, $v_r$ and $v_l$, being the left and right wheel velocities, with $\theta$ representing the angle with respect to the Earth’s $y$ axis in Figure 3.1. The UGV can be represented by its linear and angular velocities, $v$ and $\omega$ respectively, instead of the left and right wheel velocities. This is accomplished by recognizing:

\[
v = \frac{v_r + v_l}{2}
\]

\[
\omega = \frac{v_r - v_l}{l}
\]

The system then becomes:

\[
\dot{x} = -v \sin \theta
\]

\[
\dot{y} = v \cos \theta
\]

\[
\dot{\theta} = \omega
\]  

(3.2)

where $v$ and $\omega$ are now the control variables of the system. Equation (3.2) is the model used throughout for simulation of the multi-agent system.

Figure 3.1: Differential Drive UGV from Dudek and Jenkin [9].
For the following control, the UGV must be transformed into a polar coordinate system. The origin is defined as the desired trajectory point and the transformation equations are shown below.

\[ \rho = \sqrt{\Delta x^2 + \Delta y^2} \]

\[ \alpha = -\theta + \beta \]

\[ \beta = \tan^{-1} \frac{\Delta y}{\Delta x} \] (3.3)

In equation 3.3, \( \rho \) is the required distance between from the desired point, \( \alpha \) is the angle required to be facing the desired point, and \( \beta \) is the final angle to be in the correct orientation for the desired trajectory.

### 3.1.2 Control

The goal for the UGV is to move to a desired reference trajectory given by an external source. An immediate drawback to differential drive robots is the lack of lateral movement. Much like a car, a differential drive cannot move along the axis of its axle. Equation 3.4 is the mathematical expression for the particular constraint on differential drives.

\[ \dot{x} \sin \theta - \dot{y} \cos \theta = 0 \] (3.4)

Two controllers have been proposed to drive the UGV to a goal position. One controller splits the control into three steps with each controller starting after the previous is finished. The second controller transforms the dynamic equations into polar coordinates and creates a controller that removes the nonlinear aspects of the system.

### 3.1.3 Three Part Controller

For the given non-holonomic constraint, the UGV is not easily controlled through linearization and linear control methods unless a transformation of states is performed. The problem of trajectory tracking was broken into three objectives with each one solving the control law minimizing one
the of three variables in equation 3.3 at a time.

1. Moving radially until the UGV is facing the desired point, solves for $\beta$

2. Begin moving linearly until reaching the desired point, solves for $\rho$

3. Finally, move radially again until the desired orientation is reached, solves for $\alpha$

By breaking up the control system into three parts, the non-holonomic issue can be averted, and three state feedback controllers can be used for each of the state variables one at a time. Given a desired trajectory, the first step requires the UGV to reach the orientation facing the desired position before beginning its linear movement. This case requires only angular velocity, $\omega$, and our UGV becomes a single-input single-output (SISO) system corresponding to $\dot{\theta} = \omega$ with $\omega$ as the control variable. By substituting $\omega = -K_1 \beta$, where $K_1$ is the feedback gain, the UGV will successfully orient itself toward the desired trajectory. During this time, the linear velocity, $v$, is constant at 0.

Similarly, once $\beta$ has been minimized, the second part of the controller begins its computation. For the distance, $\rho$, the system becomes single-input multi-output. Now, there UGV can be represented by the first two equations of equation 3.2

$$\dot{x} = -v \sin \theta \quad \dot{y} = v \cos \theta.$$ 

Setting $v = -K_2 \rho$ will drive the UGV to the desired linear position. The same situation occurs after the desired position has been reached. The final control moves the UGV so it is facing the desired orientation. It works the same way as the first controller with a different gain value, $k_3$. Figure 3.2 shows the three states, $X$, $Y$, and $\theta$, reaching their goal coordinate, and Figure 3.3 describes the inputs sent from the controller to drive the UGV. The controllers are successful, but it is apparent the system moves in three parts. The angle moves first before the position states change, and finally the angle moves again to reach the reference angle. The next control method moves the UGV in a smooth manner and creates a linear closed loop state space form.
Figure 3.2: Response of the UGV with the three part controller

Figure 3.3: Inputs sent to UGV from the three part controller
3.1.4 Polar Transformation Controller

Based on the polar transformation of states defined in equation 3.3, the UGV can be defined with the polar coordinates as equation 3.5.

\[
\begin{bmatrix}
\dot{\rho} \\
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} =
\begin{bmatrix}
-\cos\alpha & 0 \\
\sin\alpha & -1 \\
-\sin\alpha & 0
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix}
\] (3.5)

Applying the small angle approximation, for \( \theta \ll 1 \), \( \sin\theta = \theta \), \( \cos\theta = 1 \), the system becomes

\[
\begin{bmatrix}
\dot{\rho} \\
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} =
\begin{bmatrix}
-1 & 0 \\
\frac{\alpha}{\rho} & -1 \\
-\frac{\alpha}{\rho} & 0
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix}
\] (3.6)

which is still nonlinear. A state feedback controller can be applied in place of \( v \) and \( \omega \) to create a linear system. By combining equation 3.7 and 3.6, the system will become linear and can be studied with the effects of a network applied to the system.

\[
\begin{bmatrix}
v \\
\omega
\end{bmatrix} =
\begin{bmatrix}
k_\rho & 0 & 0 \\
0 & k_\alpha & k_\beta
\end{bmatrix}
\begin{bmatrix}
\rho \\
\alpha \\
\beta
\end{bmatrix}
\] (3.7)

\[
\begin{bmatrix}
\dot{\rho} \\
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} =
\begin{bmatrix}
-k_\rho & 0 & 0 \\
0 & k_\rho - k_\alpha & -k_\beta \\
0 & -k_\rho & 0
\end{bmatrix}
\begin{bmatrix}
\rho \\
\alpha \\
\beta
\end{bmatrix}
\] (3.8)

The gain values were experimentally determined as \( k_\rho = 1.1 \), \( k_\alpha = 2.5 \), \( k_\beta = -0.75 \). The response of the UGV with this polar controller is shown in Figure 3.4. The states in this case, defined in equation 3.3, must reach 0 in order for the UGV to reach its goal destination.
Figure 3.4: Outputs of the UGV with the single polar controller

Figure 3.5: Inputs sent to the plant by the polar controller
The response shown in Figure 3.4 shows the values of $X$, $Y$, and $\theta$ by converting the polar states back to the Earth’s coordinates to show the UGV reaching its correct destination, and Figure 3.5 shows the input values required to drive towards the goal. In these figures, the system does not take any longer to reach the desired trajectory than the previous three part controller, and the inputs are driven in a more smooth manner as well lowering the jerk motion of the UGV.

For the observer based networked control issue, the observer has to be modeled first without consideration to the network. For the UGV with all states available, an observer gain matrix, $L$, is created with eigenvalues at $-5.5$, $-4$, and $-2.9$.

$$L = \begin{bmatrix} 2.9 & 0 & 0 \\ 0 & 3.6 & 0.75 \\ 0 & -1.1 & 6.0 \end{bmatrix}$$

The observer and feedback controller system was simulated with an initial observer error $[15 \ -10 \ -18]$ and to reach the state values $[-1 \ -2 \ -3]$. Figure 3.6 shows the system reaching the reference trajectory, and Figure 3.7 shows the velocities required to drive towards the reference trajectory.

**Figure 3.6:** UGV with observer based feedback and the error associated with each state
3.2 Unmanned Aerial Vehicle - Quad-Copter

Unmanned aerial vehicles, UAVs, consists of many different flying vehicles. Helicopters, quadcopters, and octocopters are some of the different UAVs currently used in research. The quadcopter will be used for modeling and control purposes. The whole goal for the quadcopter is to hover above the scene to identify a potential target for the UGV to investigate and transmit the position coordinates.

3.2.1 Mathematical Modeling

The quadcopter is a highly nonlinear system, but it can be linearized through an operating point. An approximation towards a linear system can be calculated and then a classical control algorithm, such as PID control, can be applied to the original system. A quadcopter has six degrees of freedom with 3 angular and 3 linear coordinates, $\phi$, $\theta$, $\psi$ and $x$, $y$, $z$ respectively. The quadcopter’s linear and angular velocities are defined as $p$, $q$, $r$, $u$, $v$, and $w$ respective to the previous statement of coordinates. For the quadrotor shown in Figure 3.8, the four controllable inputs are shown in equation 3.9 with thrust factor, $b$, and drag, $d$. 

![Observer Feedback Inputs](image)

**Figure 3.7:** Inputs required to drive the observer based UGV

20
\[ U_1 = T_z = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \]
\[ U_2 = \tau_\phi = b(\Omega_1^2 - \Omega_2^2) \]
\[ U_3 = \tau_\theta = b(\Omega_3^2 - \Omega_4^2) \]
\[ U_4 = \tau_\psi = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \]
\[ \Omega = \Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2 \]

(3.9)

Figure 3.8: Quadrotor representation

\( T_z \) corresponds to the thrust of the quadcopter flying vertically, while \( \tau_\phi, \tau_\theta, \) and \( \tau_\psi \) represent the torque in each of the body frame’s angles of motion. Using the control inputs, the quadcopter is well described by the following equations of motion \[11\]:

21
\[
\dot{\phi} = p + q \sin \phi \tan \theta + r \cos \phi \tan \theta \\
\dot{\theta} = q \cos \phi - r \sin \phi \\
\dot{\psi} = \frac{1}{\cos \theta} (q \sin \phi + r \cos \phi) \\
\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x} u_2 \\
\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y} u_3 \\
\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{1}{I_z} u_4 \\
\dot{u} = rv - qw - g \sin \theta \\
\dot{v} = pw - ru + g \sin \phi \cos \theta \\
\dot{w} = qu - pv + g \cos \theta \cos \phi - \frac{u_1}{m} \\
\dot{x} = w (\sin \phi \sin \psi + \cos \phi \cos \psi \sin \theta) - v (\cos \phi \sin \psi - \cos \psi \sin \phi \sin \theta) + u \cos \psi \cos \theta \\
\dot{y} = v (\cos \phi \cos \psi + \sin \phi \sin \psi \sin \theta) - w (\cos \psi \sin \phi - \cos \phi \sin \psi \sin \theta) + u \cos \theta \sin \psi \\
\dot{z} = w \cos \phi \cos \theta - u \sin \theta + v \cos \theta \sin \phi \\
\]

\[\text{(3.10)}\]

\(J_r\) and \(I_{x,y,z}\) are the rotor and body inertias and \(l\) is the length between each rotor and the body. The states of the quadcopter are defined as \(x = [\phi, \theta, \psi, p, q, r, u, v, w, x, y, z]\). Much like the UGV model, the small angles approximation is very useful to apply not only for the reductions \(\cos \theta = 1\) and \(\sin \theta = \theta\), but also so that \(\dot{\theta} = 0\) and \(\theta^2 = 0\). With just the small angle approximation, the system will still be nonlinear. Since the quadcopter only needs to be hovering over the area of interest, it can be linearized around the operating point \([0, 0, 0, 0, 0, 0, 0, 0, 0, x, y, z]\) to give the state space representation shown in equation \[\text{3.11} \]
3.2.2 Control

With the linearized system of a quadcopter operating around a hovering point described in equation (3.11), a state feedback controller can be used to control the quadcopter takeoff from rest and hover at a specified x, y, z position. A linear quadratic regulator, LQR, was used in order to find a suitable control matrix, K. With weighting matrices Q and R, the LQR goal is to minimize the cost function

\[ J(u) = \int_0^\infty (x^T Q x + u^T Ru) dt \]

and the Riccati equation

\[ A^T S + S A - (S B) R^{-1} (B^T S) + Q = 0 \]

for the matrix S. The state feedback matrix is evaluated by

\[ K = R^{-1} B^T S. \]

Figures 3.9 and 3.10 show the simulated results of the linearized quadcopter model starting from rest to reaching the final orientation. Figure 3.11 shows the values for all four inputs required to reach the final destination. Here we see the controller can successfully reach the goal destination autonomously and will be used during the simulation of multi-agent vehicles.
**Figure 3.9:** Response of a quadcopter UAV with an LQR state feedback controller

**Figure 3.10:** 3D response of the quadcopter
3.3 Wireless Communication between Agents

Communication between the mobile robots will be modeled through the TrueTime Simulink package [12]. This software environment is an additional package for Matlab/Simulink and has been used to simulate hardware components like a router, wireless network and computational delays. This versatile tool presents an opportunity to simulate in the truest sense the hardware components. This usage will remove any doubts about the accuracy of the simulation compared to the hardware. TrueTime is a Matlab/Simulink-based simulator for real-time control systems. TrueTime facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics. The communication between the two mobile agents is modeled as 802.11b WLAN, and the communication between the UGV and its controller is modeled as a wired ethernet connection.

Figure 3.11: Inputs required for control of UAV to a hovering position
Chapter 4: SIMULATION AND RESULTS

First the UGV is analyzed to determine the conservative MATI value. Next the UGV is then simulated with the model based control theory to determine considerably higher maximum values for the update times. Finally, the UAV is added to the system to send the destination trajectories for the UGV to reach.

4.1 MATI value

Equation 3.6 is the state space representation of the plant dynamics of the differential drive in polar coordinates, and all three states are available in the output. Equation 3.7 shows the controller dynamics. To solve for the MATI value, the plant and controller matrices are defined below, respectively.

\[
A_p = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_p = \begin{bmatrix} -1 & 0 \\ \frac{a}{\rho} & -1 \\ -\frac{a}{\rho} & 0 \end{bmatrix}, \quad C_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
A_c = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad C_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad D_c = \begin{bmatrix} k_\rho \rho & 0 & 0 \\ 0 & k_\alpha \alpha & k_\beta \beta \end{bmatrix}
\]

Using the plant and controller dynamics above, with just one UGV the MATI value \( \tau = 7.3414e-4 \) s. With two UGVs in the system, the MATI value becomes \( 2.4471e - 4s \). It can be seen that with more nodes in the system, the maximum interval is reduced but the values are very conservative. The value with one UGV will be used as a baseline to compare the model based control system theory.
4.2 State Feedback

For the model based network control theory, a model of the plant dynamics must be applied to
the controller. The model will reduce the network load by just sending the state values across the
communication lines. The first model used is an empty set for the model matrices shown below.

\[ \hat{A} = \emptyset_{3 \times 3} \quad \hat{B} = \emptyset_{3 \times 2} \]

With the model matrices defined, the system \( \dot{z} = \Lambda z \) can be defined for the UGV.

\[
\dot{z} = \begin{bmatrix}
1 - \frac{11}{10} h & 0 & 0 & 0 & 0 \\
0 & 1 - \frac{7}{5} h & \frac{3}{4} h & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
z \end{bmatrix}
\] (4.1)

Using the M matrix definition from equation 2.7 Figure 4.1 shows the stable region with respect
to the update time, \( h \). The UGV will successfully reach its destination as long as the update time
of the controller is less than \( \frac{1}{1.7} \) seconds.
Figure 4.1: Max eigenvalues for a UGV relative to the update time of the controller

Figures 4.2 - 4.7 show the response and input requirements for the UGV given different update times. Figures 4.2 and 4.3 have an update time of 0.1 seconds. Figures 4.4 and 4.5 have an update time of 1.0 seconds, and Figures 4.6 and 4.7 have an update time of 2.0 seconds. The simulations show the UGV can tolerate times much larger than the MATI value, $7.3414 \times 10^{-4}$ s, by 3 orders of magnitude. As long as the network update time is less than 1.5 seconds, the system should not lose stability.
Figure 4.2: State response for UGV with state feedback control and $h = 0.1\text{ s}$

Figure 4.3: Inputs for UGV with state feedback control and $h = 0.1\text{ s}$
Figure 4.4: State response for UGV with state feedback control and $h = 1.0$ s

Figure 4.5: Inputs for UGV with state feedback control and $h = 1.0$ s
Figure 4.6: State response for UGV with state feedback control and $h = 2.0 \text{ s}$

Figure 4.7: Inputs for UGV with state feedback control and $h = 2.0 \text{ s}$
4.3 Observer Based Feedback

The observer based system with the original model of the plant did not have any eigenvalues for the matrix $M$ from equation 2.9. The plant model had to be redefined. For this case a perturbation from the original model was chosen, and it is shown below.

\[
\hat{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} -0.06809 & -0.02133 \\ -0.07169 & -0.03829 \\ -0.05801 & -0.03125 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} 1.02 & 0.1 & -0.1 \\ 0.1 & 1.02 & -0.1 \\ 0.15 & 0.17 & 0.98 \end{bmatrix}, \quad \hat{D} = \begin{bmatrix} 0.02 & 0.05 \\ -0.04 & 0.1 \\ -0.035 & 0.04 \end{bmatrix}
\]

The new plant model and the observer based control increases the tolerance to higher level update time. Figure 4.8 shows the maximum eigenvalues again like in the state feedback case. Here the system can tolerate an update time around 2.4 seconds. This maximum update time is over half a second higher than the state feedback case.

![Max eigenvalue magnitude of UGV with Observer](image)

**Figure 4.8:** Max eigenvalues for a UGV relative to the update time of the controller

Figures 4.9 - 4.14 show the state responses and inputs for the observer based UGV system.
The state responses show the changes in states, the plant model response, and the error response between the observer and controller. In the inputs and error responses, the change at each update time can be seen with discrete jumps in magnitude.

Figure 4.9: Observer based system response with update time of 0.1 seconds
Figure 4.10: Observer based system with inputs from an update time of 0.1 seconds.

Figure 4.11: Observer based system response with update time of 1 second
Figure 4.12: Observer based system with inputs from an update time of 1 second

Figure 4.13: Observer based system response with update time of 2.5 seconds
4.4 State Feedback with Network

Networks typically have time delays associated with sending information through communication lines. This can be interpreted separately from the update time and complicating the system response as shown in equation 2.12. The plant model is the same perturbed system as in the observer case. Figure 4.15 shows the max eigenvalues with respect to both the update time and time delay associated with the network transmission.

Figure 4.14: Observer based system with inputs from an update time of 2.5 seconds
Two separate network delay times were chosen for simulation, $\tau = 0.1$ s and $\tau = 0.6$ s. For $\tau = 0.1$ s, Figures 4.16 - 4.20 show one stable and one unstable state response and inputs. For the smaller network delay of 100 ms, the stable system, Figures 4.17 and 4.18, reaches the destination within 15 seconds, a reasonable time. It can be seen that with the increased time delay, the system takes substantially longer to reach its destination.

**Figure 4.16:** Max eigenvalues for a UGV versus update time of controller with $\tau = 0.01$ s
Figure 4.17: State response for UGV with network delay = 0.1 s and update time = 1 s

Figure 4.18: Inputs for UGV with network delay = 0.1 s and update time = 1 s
Figure 4.19: State response for UGV with network delay = 0.1 s and update time = 2.01 s

Figure 4.20: Inputs for UGV with network delay = 0.1 s and update time = 2.01 s

Figure 4.21 shows the max eigenvalues and subsequently the stability of the system with a
network delay of 600 ms. There is noticeably less stable area with this network delay. The system can only have an update time roughly between 0.7 and 0.85 seconds. Figures 4.22 - 4.25 show the response and inputs for two different update times. The stable system takes over 50 seconds to reach marginal stability, over 2.5 times as long as the system with only 100 ms delay.

![Network Delay Stability](image)

**Figure 4.21:** Max eigenvalues for a UGV versus update time of controller with $\tau = 0.6$ s
Figure 4.22: State response for UGV with network delay = 0.6 s and update time = 0.7 s

Figure 4.23: Inputs for UGV with network delay = 0.6 s and update time = 0.7 s
Figure 4.24: State response for UGV with network delay = 0.6 s and update time = 1.5 s

Figure 4.25: Inputs for UGV with network delay = 0.6 s and update time = 1.5 s
4.5 UAV and UGV system

With the UGV modeled with an update time and network delays, the system consisting of the UAV and UGV can be simulated to verify the update times for stability given another mobile platform added into the mix. The setup of the system with two robots is shown in Figure 4.26. Figures 4.27 - 4.30 depict the movement of the UGV which is given its destination coordinates by the UAV. Subsequently, the UGV sends a signal back to the UAV when it requires another destination. This system communication is modeled through a wireless communication module between the two vehicles. As stated before, there are three different time delays associated with the system, and five different models were tested where different delays were considered.

- Wireless network delay between vehicles only
- Ethernet based delay within the UGV only
- Computational Delay for UGV controller only
The simulation is set up in such a way that data is only sent when needed from the UAV to UGV. Therefore, the wireless network is not saturated with information continuously and the delay associated is so small the system models very close to the original simulation without time delays. In Figures 4.27 and 4.28 the dotted lines show the effect of just the wireless communication delay, which is very similar to the simulation with no delays. When the wireless network is not being continuously used, the delay propagation is small.

When communication delay for the UGV from its controller to the plant is considered, also shown in Figures 4.27 and 4.28, the system reaches the trajectory faster, but the movement is much choppier. Interestingly, when the communication delay is small much like in the first results, the new input magnitudes lag and the higher magnitudes move the UGV closer than anticipated. This results in it reaching the desired trajectory faster. An issue will occur if the delays are too big and the system begins to overshoot the goal.

For the first two delays separately, there is not a big change compared to the delay-less system. These systems, however can be pushed to instability with higher delays in either system.

Figures 4.29 and 4.30 depict two systems, one with the computational delay within the UGV’s controller and another with all three of the previous delays together.

The computational delay was stated to be 0.5 seconds long for the UGV controller, which again shows the same effect as the communication delay inside the UGV. The system moves faster to the position, but the angular movement oscillates and doesn’t reach its desired point. The computational delay creates an instability in the controller at 0.5 seconds.

The final simulation results stem from the system with all three of the previous delays included. The simulation here should mimic a real-time system the closest. For this system, the trajectory of the UGV takes longer to begin moving and travels less smoothly than the previous simulations. While the UGV still reaches the desired trajectory, it does not reach the required angular orientation. The oscillations increase much more than the system with just a computational delay.
Figure 4.27: UGV & UAV simulation with delays in the X coordinate of UGV

Figure 4.28: UGV & UAV simulation showing angular motion for differing delays

For the system overall, we can conclude the delays associated with a 1 Gbps wired connection will decrease the settling time of a system when used continuously. A 50 Kbps wireless network
affects the system minimally when only used to transmit data sparsely when needed, but it can be postulated to degrade the system considerably if used continuously. With all three delays, the system will still reach the desired endpoint with more time and less smooth movement.

Figure 4.29: UGV & UAV simulation with delays showing the X coordinate of UGV
Figure 4.30: UGV & UAV simulation with delays showing angular movement of the UGV
Chapter 5: CONCLUSIONS AND FUTURE DIRECTIONS
BIBLIOGRAPHY


VITA

Nicholas Jay Quinterro Gamez was born on the 19th of March 1992 in Cuero, Texas. He received his B.A in Chemistry with minors in Mathematics and Physics from Texas A&M University in College Station. After deciding a change of pace was needed, he is currently studying for a M.S. in Electrical Engineering at The University of Texas at San Antonio. He plans to work in areas either related to robotic control systems or machine learning.