Worcester Polytechnic Institute

M.S. Robotics Engineering Capstone Report

High speed robotic convoying over rough terrain

Author:
Aaron Fineman

Advisor:
Professor Stephen Nestinger

May 4, 2012

Abstract

The goal of this project is to apply various high level control algorithms, in particular potential field based methods, for use in robotic convoys. With a priori knowledge of the kinetic models of the other robots in the convoy, the following robots will be able to determine the intended path and trajectory of their leader without the use of explicit communication. Kinematic and dynamic models were created for the robots. This allowed the simulation of movement over rough terrain. The high level control algorithms was implemented on several Khepera III platforms. The convoy used the robot model to traverse over simulated rough terrain.

Contents

List of Figures

Chapter 1

Introduction

This project was inspired by searching for ways to control full-scale convoys. This application is commonly seen in military convoy driving, where a line of vehicles must follow each other while maintaining a safe and constant distance. For smaller vehicles, the advantage is the ability to gauge the path ahead, and avoid obstacles that the preceding vehicle failed to detect. In larger convoys, if the leader malfunctions, the convoy may continue unimpeded, rather than become trapped waiting for the leader to continue. However, there are plenty other applications including, but not limited to, transportation inside a factory.

Chapter 2

Related work

2.1 Convoying

Robotic convoying is a significant research problem in mobile robotics. This problem has been addressed from different points of view, e.g., artificial vision, control theory, artificial intelligence, etc. [2]. Most existing mobile robot systems still involve a single robot working alone, while there are a wide range of potential applications for robots acting in concert [2].

The authors of [3] discuss how a following robot imitates its lead robot, where each of the following robots track the angular and linear velocity of its lead robot. They also presented how to convoy with constant distance between robots while analyzing the case as Velocity Pursuit, Deviated Pursuit, and Proportional Navigation.

2.2 Potential Fields

The study of groups of multiple autonomous mobile robots has been of great interest [4]. In high-speed convoying, fast and efficient coordinated maneuvers are a must, as well as quickly processing data to make a decision. Biologists have observed remarkable group-level characteristics in animal aggregations such as swarms, flocks, school and herds [4]. Some of these characteristic are the ability to make very fast and efficient coordinated maneuvers, quickly process data and to cooperatively make decisions. Thus, there is a significant interest within the robotic community, to better understand the biological imperative and exploit it by incorporating similar principles in artificial robot collectives.

Researchers typically refer to the use of potential field as a means for assisting a robot to move from one initial configuration to a desired final configuration without colliding with any obstacles. Potential fields consist of force vectors, caused by the obstacles or target positions, which may be linear or tangential and they may have characteristics of repulsive, attractive or random, depending on the state of the agent with respect to its environment [5]. It was originally

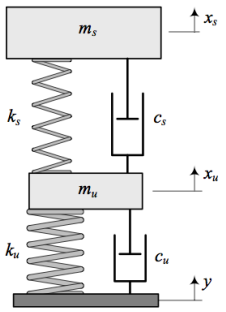

Figure 2.1: A generalized quarter car model with damped tires [1].

developed as an on-line collision avoidance approach, applicable when the robot does not have a prior model of the obstacles, but rather senses them during motion execution [6]. The most common potential field function is defined as:

$$F(q) = -\nabla(U_a(q) + U_r(q)) \tag{2.1}$$

It is defined over free space as the sum of an U_a attractive potential pulling the robot toward the goal configuration and a U_r repulsive potential pushing the robot away from the obstacles [7]. This model attempts to find a collision-free, feasible, path in the neighborhood of the initial estimate. The goal is to define a path for the robot to follow, starting from the initial position moving towards the end/goal position calculated by the negative gradient of the entire field. If the start and the goal configurations cannot be connected through a sequence of feasible configurations, the initial path is then assumed not to lead to a solution. The potential field method has increased in popularity for the control of mobile robots, due to its mathematical simplicity [7].

2.3 Car models

In order to study the dynamics of a vehicle system, these are typically modeled as a quarter car, a half car, and a full car.

2.3.1 Quarter car

To model a car, the typical approach is to break it down into smaller parts, and then recombine the parts at the end. To begin with, a quarter car model is created, showing only one wheel, constrained to one dimension (vertical displacement,) and the associated suspension [2, 4]. The quarter car model is shown in figure 2.1. The car body is represented by the mass M and is connected to a suspension system consisting of a spring K and a damper B . The suspension is connected to the tire which is modeled as a mass M and a spring K connecting it to the road.

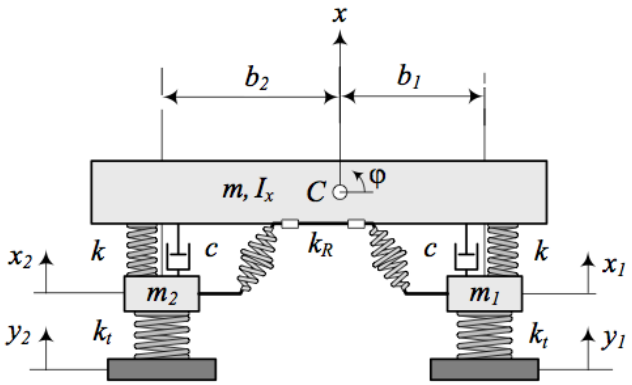

Figure 2.2: A generalized half car model with a roll-bar [1].

2.3.2 Half car

The half car model represents a two-dimensional view of the vehicle head-on as shown in figure 2.2. Figure 2.3 shows the half car model overlaid on a car body as reference. It is actuated in two dimensions, vertical displacement and rotation about the center of gravity. Horizontal motion (slip) may also be shown on this model. As can be seen from the figure, the half car model is composed of two quarter car models acting on a common strut to realize roll.

2.3.3 Full car

Full Car. The full car model represents a three-dimensional view of the vehicle as shown in figure 2.4. Figure 2.5 shows the full car model overlaid on a car body as reference. The full car model is composed of two half car models, one representing each side (front and back.) From these models one gets a combined roll and pitch. There is then a third (top down) view added to the model that allows for the calculation of yaw, and factors the input velocity into the model.

2.3.4 Steering control

Cars are steered by an Ackermann drive system. However this can be tedious to model, so a bicycle model of the steering is often adopted [8, 9, 10]. The bicycle control law is simple, and geometrically intuitive [8]. A kinematic bicycle model is described by:

$$\dot{x} = v\cos\theta \tag{2.2}$$

$$\dot{y} = v\sin\theta \tag{2.3}$$

$$\dot{\theta} = \frac{v}{L}\tan\delta \tag{2.4}$$

$$\tag{2.5}$$

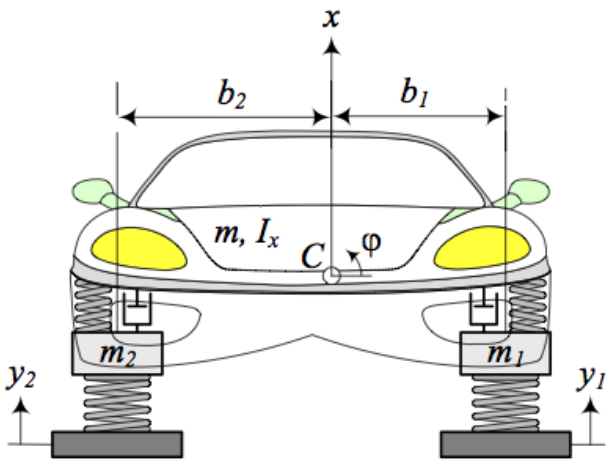

Figure 2.3: A generalized half car model overlaid on a car body [1].

x, y, and θ are with respect to the robot. v is the speed of the robot, and L is the wheelbase length. δ is the turning angle of the wheels in the robot's coordinate frame.

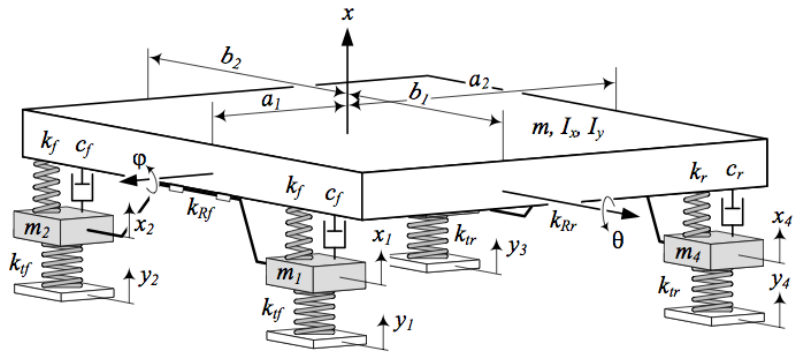

Figure 2.4: A generalized full car model with roll-bars [1].

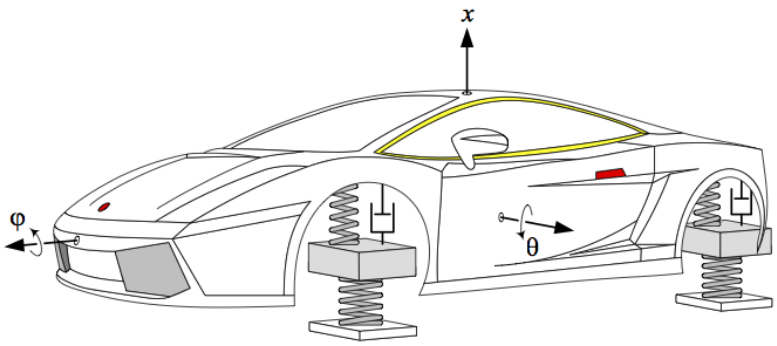

Figure 2.5: A generalized full car model overlaid on a car body [1].

Chapter 3

Methodology

3.1 Potential fields

The following algorithm to control a convoy was developed and implemented on three Khepera III robots. The trajectory of motion was generated using a version of the potential fields algorithm described in the previous section. The aim was to design a control law for n number of unconnected wheeled autonomous robots to follow a lead robot's trajectory while maintaining a constant distance from each other. The lead robot's trajectory will be generated from the potential field algorithm. The control algorithm utilizes the potential field force equation (2.1) to create a matrix of possible configurations for the movement of the robot. The robots' trajectories are defined by the following control equations:

$$q_n = q + -K * F(q)q \tag{3.1}$$

$$q_{n(r)} = q_{(r+1)} - o \tag{3.2}$$

Equation (3.1) determines the lead robots next position (q_n) based on its current position (q) and the potential field force equation (2.1). The position q and q_n are 2x1 matrices of the form $[xy]^T$. The following robots' new positions ($q_{n(r)}$) are determined by equation (3.2) for a simple leader-follower simulation. The offset from the lead robot ($q_{(r+1)}$) is o, a 2x1 matrix of the form $[xy]^T$ representing the straight-line x and y distance between the two robots.

In addition to a simple leader-follower simulation, a second simulation with a smoothing algorithm was implemented in MATLAB. The smoothing control algorithm accomplishes two goals; first, if a following robots leader disappears (i.e. breaks down) the follower becomes the leader and reaches its goal. Second, the following robots are able to follow a smoother trajectory to the goal. A smoother trajectory has less radical turns and directional adjustments required by the robot. The more follower robots, the smoother the final trajectory becomes. This algorithm uses these two equations for the follower robots:

$$F(q) = \begin{cases} r = 1, & -\nabla(U_a(q) + U_r(q) \\ r > 1, & -\nabla(q_{(r+1)} + U_r(q) \end{cases} \tag{3.3}$$

$$q_{n(r)} = q_r + -K * F(q)\dot{q} \tag{3.4}$$

Equation (3.3) uses the same potential field algorithm as the other simulation except for follower robots, where instead of the attractive field being assigned by the goal position; the attraction is defined by the robot's respective leader's position. For each time step, the robot's leader's position becomes the attractive force (3.3) and a new position $q_{n(r)}$ is calculated based on it (3.4). Using this method, the following robots follow smoother trajectories because they react to the information learned by their leader about the terrain. It also solves the issue of a leader robot breaking down; the following robot becomes the new leader and successfully reaches the goal.

The Khepera III's that were used to implement the potential fields simulation use the standard differential drive method. The kinematic equations for this type of robot are:

$$\dot{x_i} = v_i(t)\cos\theta_i \tag{3.5}$$
$$\dot{y_i} = v_i(t)\sin\theta_i \tag{3.6}$$
$$\dot{\theta_i} = \omega_i(t) \tag{3.7}$$

where (x_i, y_i) are the coordinates of the reference point of i-th robot in the Cartesian frame of reference. θ_i is its orientation angle with respect to the positive x-axis. v_i and $\omega_i(t)$ are the linear and angular velocities, respectively.

3.2 Robot tracking

To calculate the current location of the robot in the world based off the current ground elevation and/or disturbances, the dynamics of the system are required. The dynamics of car based systems take on the following characteristic form:

$$[M]\ddot{x} + [b]\dot{x} + [k]x = F \tag{3.8}$$

where $[M]$ are the mass coefficients, $[b]$ are the damping coefficients, and $[k]$ are the spring coefficients. The full realization of this characteristic equation may be seen in appendix A.1.

Knowing the position and changes in the center of mass of the vehicle, equation (3.8) may also be used to calculate the inverse dynamics of the system. This allows for a following vehicle to be able to calculate the ground input on the leading vehicle based off it's movements.

Turning using the bicycle model from equation (2.2) was implemented on the robots in the simulation. The bicycle model was backed by a simple turn right controller. If the following vehicle detected a disturbance higher than it's threshold value, it attempted to turn right one vehicle width plus a buffer before it hit the obstacle. If there is not enough room available to the right, the controller falls back to the left.

Chapter 4

Implementation

4.1 Potential fields

The potential fields control algorithm simulation was implemented inside MATLAB, as described above. It is capable of successfully leading n-number of robots from the start position to the goal, assuming that the wheeled mobile robots move in the horizontal plane. Due to hardware limitations, the Khepera robots were only used to simulate the potential field control algorithm. The robot's lack the ability to traverse rough terrain, therefore the high speed convoying and rough terrain navigation was simulated through MATLAB. A screen capture of the algorithm in progress can be seen in figure 4.1. Three robots are shown avoiding a simple obstacle (in this particular situation, a chair.) The background shows the gradient representation of the goal potential field (located slightly north of the chair.) Blue represents the lowest areas of the gradient, and red the highest. In this particular run, the objects have a very narrow and tall field around them.

4.2 Dynamics

A simulation of the robot was created in MATLAB using the derived dynamics. A terrain was given as input, and the robot was simulated driving over it, returning the state of it's centers of mass. To calculate the state, the ode45 solver in MATLAB was used to iterate over the full version of the characteristic equation (3.8). This gave access to the full state of the centers of mass, namely the accelerations, velocities, and positions of the body center of mass, tire centers of mass, and the body angles.

Following this, a second robot was simulated driving a fixed distance behind the first one and matching the speed, as shown in figure 4.2. It recorded the changes in position, velocity, and acceleration of the body center of mass of the vehicle in front. Because the system is not rigid, the dynamics depend on the position of the wheels. However, this data would not be able to be

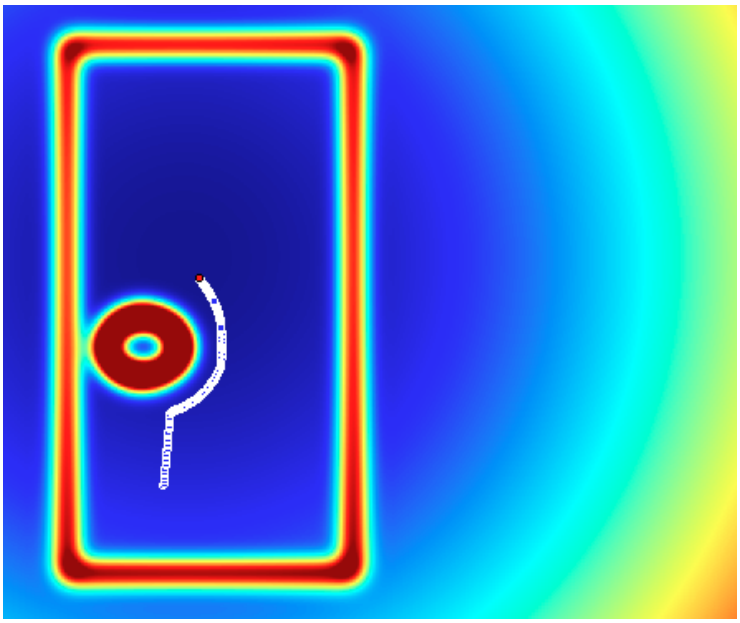

Figure 4.1: A screen capture of the potential fields control simulation

recorded outside of a simulation. To account for this, the following robot runs the dynamics on the last state of the lead robot in an attempt to guess the current location of the wheels. The estimated wheel states from the last time step are combined with the real body states from the current time step to create an estimation of the change in ground height under the lead vehicle.

A steering model was then implemented on the robots. The first robot had it's steering fixed to ensure it would drive directly over the obstacle. The following robots then attempted to move right one vehicle width plus a buffer space from the center of the preceding robot. The use of only one vehicle width was chosen as it is assumed that the disturbance the first vehicle hit was small enough for it to not be picked up by sensors, but larger than prudent to blindly drive over (such as potholes.)

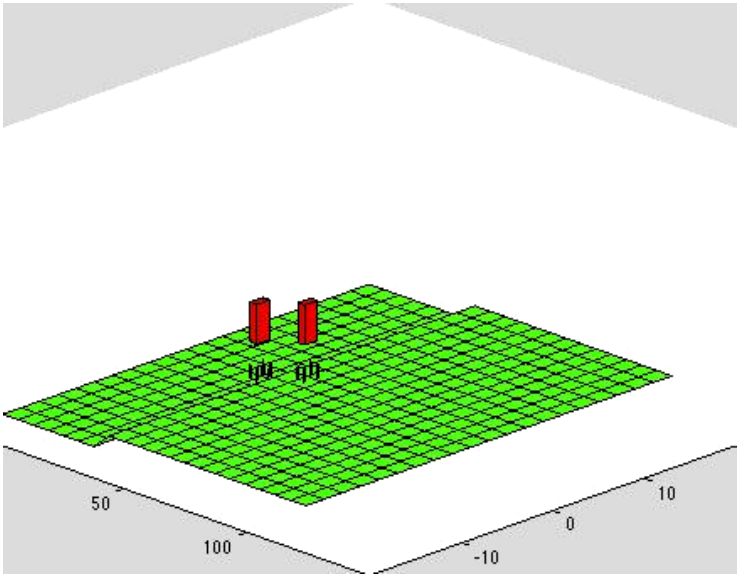

Figure 4.2: A screen capture of the convoy approaching a bump.

Chapter 5

Results

5.1 Potential fields

The MATLAB simulations proved the validity of the potential field control algorithm, as an n-number of robot convoys successfully followed the leader's trajectory, while maintaining a constant distance from each other. In addition, the smoothing control algorithm was also proven during simulation, as the following robots were able to follow a smoother trajectory to the goal, resulting in less radical turns and directional adjustments. In a more realistic situation, if a following robots leader disappears (i.e. breaks down) the follower becomes the leader and reaches its goal.

5.2 Dynamics

Running each of the dynamic models (quarter car, half car, full car,) produced a set of graphs detailing the system response. These showed the system traversing several terrain types, but for the clarity of this report, only the system traversing a 2cm bump directly was included (more thorough results can be found in appendix B.1). This is one of the worse-case scenarios the system would have to deal with, and produces the clearest graphs. Once the simulation is run, the car is given 3 seconds or flat terrain for the system to settle from its 0-configuration before it hits the 2cm bump. In the first 3 seconds, the system can be clearly seen undergoing a damped oscillation to its equilibrium position. The half car and full car models also show the roll and pitch of the system respectively. For this simulation, the geometry was defined such that there would be no lateral or horizontal displacement. At the 3 second mark, the system can be seen traversing the bump. This is most pronounced in the quarter car model, as the shock and damper system absorb the majority of the body displacement in the half car and full car models. The effect of the bump can still be seen on the wheels. The bump can also be seen effecting a pitch in the full car model. As the front wheels hit the bump, the car pitches upwards until the back wheels hit

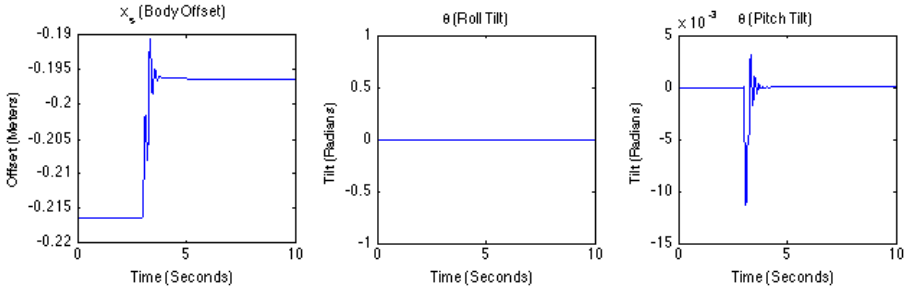

Figure 5.1: The body response of a car traversing a 2cm bump head on.

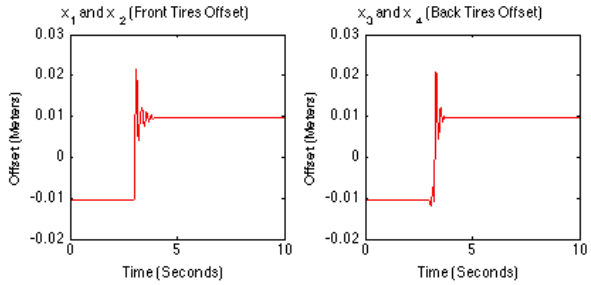

Figure 5.2: The wheel response of a car traversing a 2cm bump head on.

the bump causing a matched downward pitch. This induces some oscillations which become damped quickly.

Running concurrently with the dynamics are the inverse dynamics. These are calculated from the point of view of a system following the leader. The inverse dynamic model is able to correctly predict the sag of the system from its 0-configuration within 1% error. For most of the terrain we generated, the inverse dynamics were able to correctly predict the response of the system within 2%-5% error. These large bumps are the most extreme case we were able to find. They exhibit up to 10% instantaneous error; however they drop back to a baseline 1% error within 0.2 seconds. This error is a result of the naive approach taken with calculating the wheel states using the last time step. If the states were filtered, the estimates are even closer. However, there was not time to create a rigorous filter for this purpose.

The steering control performed acceptably. Because of the accuracy of the inverse dynamics, the steering control was reduced to a function of the vehicle's turning radius and turning speed. When running the simulation at 13 m/s (approximately 30 mph,) the following vehicles were able to avoid the obstacle within 1.5 seconds. Based off the car values for the simulation (outlined in appendix A.2,) a following distance of 25 meters is optimal for this situation.

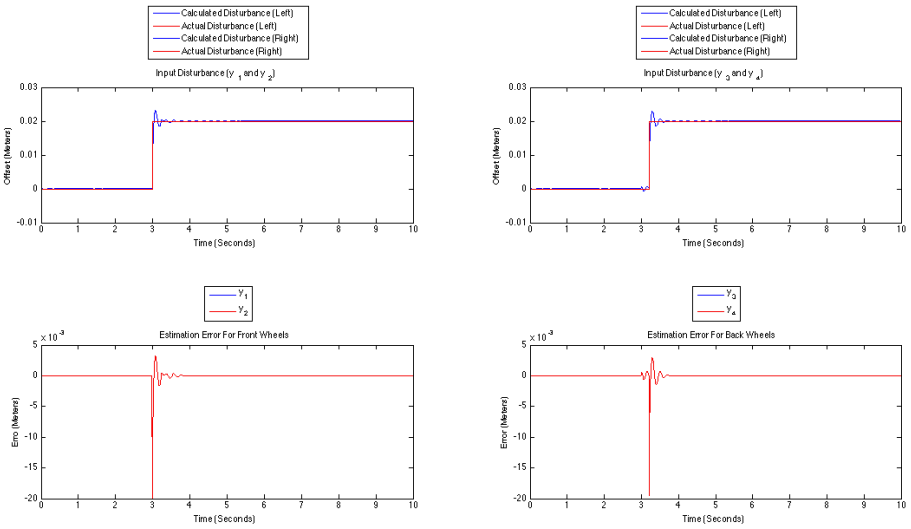

Figure 5.3: The estimation of a car traversing a 2cm bump head on.

Chapter 6

Conclusion

This document outlined the development of a full control problem of a convoy of high speed robots over rough terrain. It presented the use of a Potential Fields algorithm to control the convoy, while generating a trajectory reference for the robots to follow. Assuming the availability of a priori kinetic models of the other vehicles in the convoy, the followers were able to determine the intended trajectory of their leader without the use of explicit communication.

6.1 Future work

As a continuation of this project, this work should be taken outside. R.C. cars appear to be a suitable device for a proper proof of concept. In particular, the Traxxas Stampede line[1] looks appealing, and several areas of this project were completed with this in mind. However, working outdoors brings a host of other concerns that were not addressed in this report. These are smaller platforms, and as such are not damped as much as a full sedan car. This will add a lot of noise into the system, even if it is modelable. In addition, There will be severe sensor saturation while in sunlight (even indirect.)

[1] `http://traxxas.com/products/models/electric/3605stampede`

Acknowledgment

I would like to greatly thank Professor Stephen Nestinger for advising me during the course of this research. I would also like to thank Michael Audi, Michael Raineri, and Annette Rivera for their assistance, and for keeping me focused.

Bibliography

[1] R. N. Jazar, *Vehicle Dynamics: Theory and Application.* Springer, 2008.

[2] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Experiments in sensing and communication for robot convoy navigation," *Intelligent Robots and Systems Human Robot Interaction and Cooperative Robots*, 1995.

[3] F. Belkhouche and B. Belkhouche, "Modeling and controlling a robotic convoy using guidance laws strategies," *IEEE Transactions on Systems, Man, and Cybernetics*, 2005.

[4] L.-F. Lee, "Decentralized motion planning within an artificial potential framework (apf) for cooperative payload transport by multi-robot collectives," *State University of New York at Buffalo*, 2004.

[5] K. F. Uyanik, "A study on artificial potential fields," *Middle East Technical Univ.*

[6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," 1986.

[7] M. E. Vladimir Milic, Josip Kasac, "A potential field method approach to robotic convoy obstacle avoidance," *Annals of DAAAM and Proceedings*, 2009.

[8] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, 2009.

[9] M. A. Sotelo, "Lateral control strategy for autonomous steering of ackerman-like vehicles," *Robotics and Autonomous Systems*, 2003.

[10] S. F. Campbell, "Steering control of an autonomous ground vehicle with application to the DARPA urban challenge," *University of Notre Dame*, 2005.

Appendix A

Dynamics

A.1 Car model derivation

The dynamics for a simple car takes on the characteristic form shown in equation (3.8):

$$[M]\ddot{x} + [b]\dot{x} + [k]x = F \tag{A.1}$$

However, the full (populated) dynamic models are significantly more complicated, and reproduced in full below.

Notation	Description
mu	Unsprung mass; body mass
ms	Sprung mass; wheel mass
mf/mr	Front/rear tire mass
Ix	X Inertia; roll
Iy	Y Inertia; pitch
ku	Unsprung spring; suspension spring constant
ks	Sprung spring; tire spring constant
kf/kr	Front/rear suspension spring constant
ktf/ktr	Front/rear tire spring constant
bu/cu	Unsprung damper; suspension damping
bs/cs	Sprung damper; tire damping constant
cf/cr	Front/rear suspension damping constant
a1/a2	Distance from the center of mass to the front/rear wheels
b1/b2	Distance from the center of mass to the left/right wheels

A.1.1 Quarter car

$$[M] = \begin{bmatrix} ms & 0 \\ 0 & mu \end{bmatrix}$$

$$[b] = \begin{bmatrix} bs & -bs \\ -bs & (bs + bu) \end{bmatrix}$$

$$[k] = \begin{bmatrix} ks & -ks \\ -ks & (ks + ku) \end{bmatrix}$$

A.1.2 Half car

$$[M] = \begin{bmatrix} ms & 0 & 0 & 0 \\ 0 & Ix & 0 & 0 \\ 0 & 0 & mf & 0 \\ 0 & 0 & 0 & mf \end{bmatrix}$$

$$[b] = \begin{bmatrix} 2 * bu & (bu * b1 - bu * b2) & -bu & -bu \\ (bu * b1 - bu * b2) & (bu * b1^2 + bu * b2^2) & -bu * b1 & bu * b2 \\ -bu & -bu * b1 & bu & 0 \\ -bu & bu * b2 & 0 & bu \end{bmatrix}$$

$$[k] = \begin{bmatrix} 2 * ku & (ku * b1 - ku * b2) & -ku & -ku \\ (ku * b1 - ku * b2) & (ku * b1^2 + ku * b2^2) & -ku * b1 & ku * b2 \\ -ku & -ku * b1 & (ku + ku) & 0 \\ -ku & ku * b2 & 0 & ku * ku \end{bmatrix}$$

A.1.3 Full car

$$[M] = \begin{bmatrix} mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Ix & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Iy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & mf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & mf & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & mr & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & mr \end{bmatrix}$$

$$[c] = \begin{bmatrix} C11 & C12 & C13 & -cf & -cf & -cr & -cr \\ C21 & C22 & C23 & -b1 * cf & b2 * cf & b1 * cr & -b2 * cr \\ C31 & C32 & C33 & a1 * cf & a1 * cf & -a2 * cr & -a2 * cr \\ -cf & -b1 * cf & a1 * cf & cf & 0 & 0 & 0 \\ -cf & b2 * cf & a1 * cf & 0 & cf & 0 & 0 \\ -cr & b1 * cr & -a2 * cr & 0 & 0 & cr & 0 \\ -cr & -b2 * cr & -a2 * cr & 0 & 0 & 0 & cr \end{bmatrix}$$

Element	$Value$
C11	$2 * (cf + cr)$
C12	$(b1 - b2) * cf - (b1 + b2) * cr$
C13	$2 * (a2 * cr - a1 * cf)$
C21	$C12$
C22	$(b1^2 + b2^2) * cf + (b1^2 + b2^2) * cr$
C23	$(b2 - b1) * a1 * cf - (b1 + b2) * a2 * cr$
C31	$C13$
C32	$C23$
C33	$2 * cr * (a1^2 + a2^2)$

$$[k] = \begin{bmatrix} mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Ix & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Iy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & mf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & mf & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & mr & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & mr \end{bmatrix}$$

A.2 Car model parameter values

The simulation was tested with several different car parameters from normal sedans and buses to R.C. trucks. However, most of the work was done using the parameters for the sedan found in Vehicle Dynamics [1]. The specific values used are reproduced below. The values in parenthesis describe a more full car without the geometric center and center of mass co-located.

A.2.1 Mass values

Object	Mass (kg)
Body mass	840
Front wheel mass	53
Rear wheel mass	53 (76)
Roll inertia	820
Pitch inertia	1100

A.2.2 Vehicle lengths

Object	Length (m)
Center of mass to front wheel	1.4
Center of mass to rear wheel	1.4 (1.47)
Center of mass to left wheel	0.7 (0.75)

A.2.3 Spring values

Object	Stiffness (N/m)
Front suspension	10,000
Rear suspension	10,000 (13.000)
Front tire	200,000
Rear tire	200,000

A.2.4 Damping values

Object	Damping (N*s/m)
Front suspension	9,600
Rear suspension	9,600
Tires	0

A.2.5 Miscellaneous values

Object	Value
Maximum speed	13 m/s

Appendix B

System response

B.1 Dynamics

Following are figures detailing the complete system response of the full car dynamic model to two different situations. The first situation (figure B.1) is the vehicle driving directly into a raised 2cm bump. As can be seen there is no roll created in this system. In the second situation (figure B.2) the vehicle drives over the side of the same obstacle so one side remains on the ground and the other drives directly up the 2cm bump. In this system, the roll can be clearly seen.

B.2 Inverse dynamics

Following are the figures detailing the complete system response of a full car as viewed by a following vehicle. The lead vehicle executes the same two maneuvers as in appendix B.1, and the trailing car follows and records. As in figure B.1, figure B.3 exhibits no roll as the vehicle drives directly over the bump. However, figure B.4 clearly shows the roll that the leading car is experiencing as it drives over the edge of the 2cm bump.

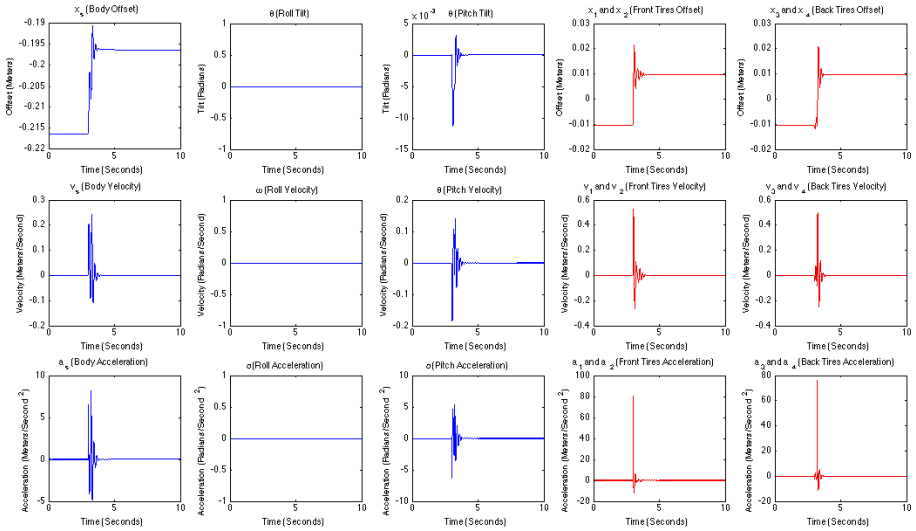

Figure B.1: The dynamics response of a full car driving directly into a 2cm bump.

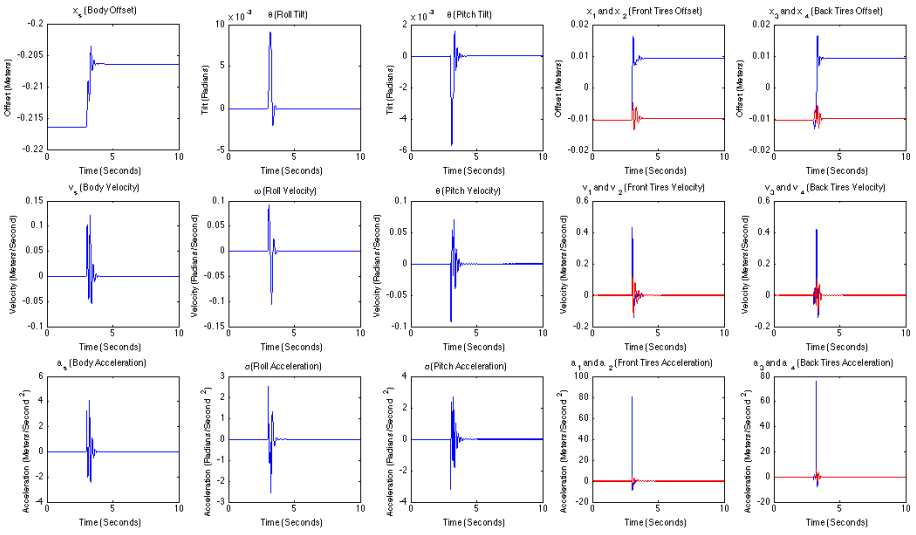

Figure B.2: The dynamics response of a full car driving into the side of a 2cm bump.

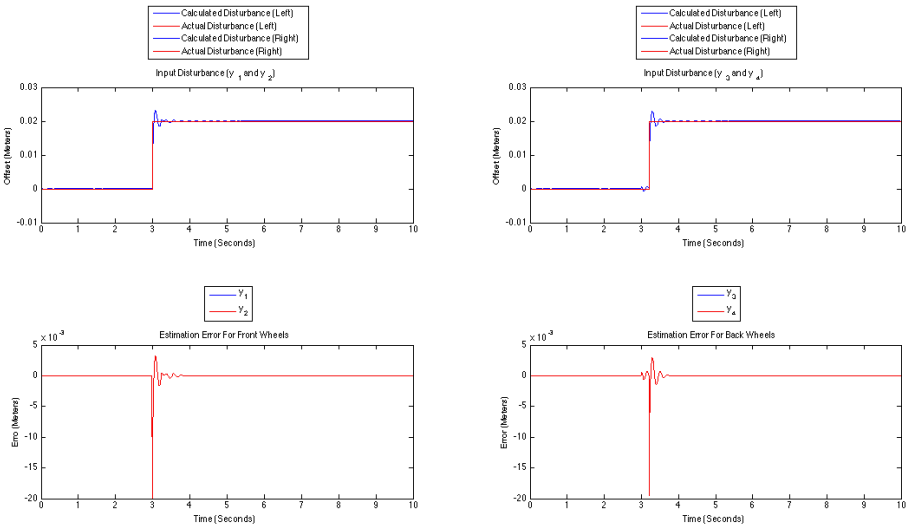

Figure B.3: The inverse dynamics response of a full car driving directly into a 2cm bump.

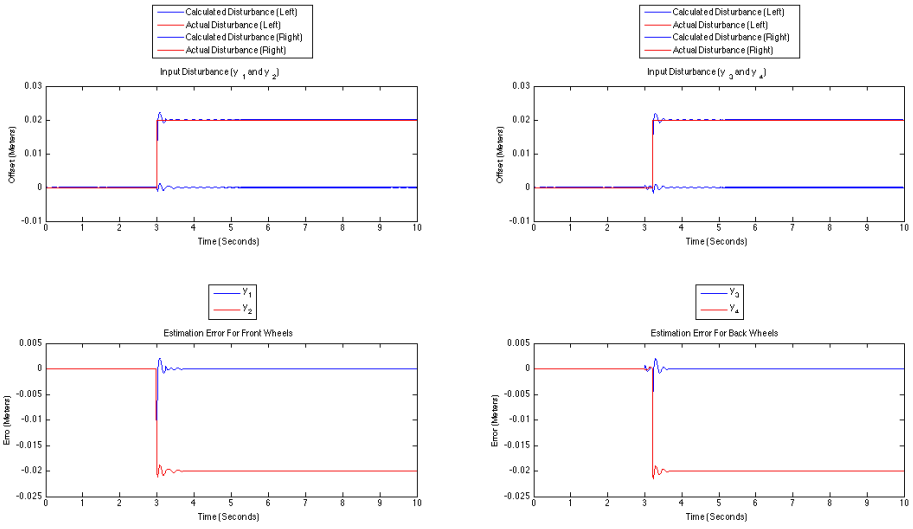

Figure B.4: The inverse dynamics response of a full car driving into the side of a 2cm bump.

Appendix C

Working with Khepera III's over serial

Following are some miscellaneous notes on working with the Khepera III robots via a bluetooth serial connection. Khepera III robots with a KoreBot II board disable the direct Khepera III bluetooth to serial connection and instead expose serial access to the on-board Linux distribution via the bluetooth connection. The Khepera III's (from here on, the use of the KoreBot II module is assumed) identify themselves as "KHIII id" over bluetooth. They can connect with any passcode ("0000" was used,) however the connection is slow to initialize, so two consecutive attempts may be needed with different passcodes. Once connected, they are able to provide a standard serial device (/dev/tty.KHIIIid-BluetoothSer on Mac OSX.) Picocom[1] is known to work with the default (9600) baud rate, however higher baud rates are supported.

ASCII files can be transfered seamlessly via 'cat.' Once connected with a serial terminal, the command "cat >> filename << EOF" on the Khepera will listen until an end of file (EOF) character is recieved. On the connected computer, piping a file to the device will transfer it ("cat filename /dev/tty.KHIIIid-BluetoothSer" on MacOSX.) Once the file is transfered, end of file (ctrl-d) can be manually entered on the Khepera to close and save the file.

However, binary files (such as .p12 encrypted wireless certificates) cannot be transfered via this method. When interpreted in ASCII mode, the binary is seen as control sequences which are executed, causing this method to fail. The Khepera provides tools for use with the zmodem transmission protocol which may be used. These tools are rarely installed on modern computers, and can generally be provided by the lrzsz[2] package on most Linux distributions and Mac OSX. Picocom integrates with these tools automatically, so once installed, binary files may be transmitted from the computer to the connected Khepera. The default key combination in Picocom is C-a C-r.

[1] `http://code.google.com/p/picocom/`
[2] `http://ohse.de/uwe/software/lrzsz.html`

As a note, wpa_supplicant does not provide an init script, so it must be started manually, or via a user provided init script, such as the one below. The dhcp client, udhcpc also does not start automatically.

```
#!/bin/sh
wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf -B
udhcpc
# end of script
```

MATLAB makes it very easy to work with the Khepera's over serial. The following code will create a serial object and open it, then log in and execute a program.

```
s = serial('/dev/tty.KHIII13914-BluetoothSer');
fopen(s);
fprintf(s,'root');
fprintf(s,''); % Blank password
fprintf(s,'./khepera3_test');
```

Sending text output and commands is as simple as using fprintf. The following code demonstrates generating an output string controlling the wheel speed, then sending it to the running Khepera demo program. The use of the C-styled 0-decimal length floating point output simulates sending integers.

```
output_string = sprintf('setmotspeed %.0f %.0f', speed_l, speed_r);
fprintf(s,output_string);
```

The serial object is also easy to close and clean up, as demonstrated by the following code.

```
% Write 0 to the robot and create a clean state
output_string = sprintf('setmotspeed 0 0');
fprintf(s,output_string);
output_string = sprintf('exit');
fprintf(s,output_string);
output_string = sprintf('exit');
fprintf(s,output_string);
fclose(s);
delete(s);
clear s;
```

If the bluetooth serial connection fails, or the program exits uncleanly, the remaining serial objects are also easy to gather up and close, either manually or automatically.

```
try
   out1 = instrfind;
   fclose(out1);
catch error

end
```