Real-time Safe Semi-Autonomous Control with Driver Modeling

Victor Shia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-41
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-41.html

May 1, 2014

Acknowledgement

This material is based upon work supported by the National Science Foundation under Awards ECCS-0931437, ECCS-1239323, and DGE-0903711. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Abstract

Real-time Safe Semi-Autonomous Control with Driver Modeling

by

Victor Andrew Shia

Distracted driving is still a major concern on roadways today. According to the US Department of Transportation, in 2011, 387,000 people were injured in crashes involving a distracted driver. While there has been tremendous work in vehicular safety systems, as the velocity of the vehicle increases, threat assessment becomes more difficult to predict. In this regime, understanding how the driver behaves is critical in determining the safety of the vehicle. This work presents a framework for incorporating human driver modeling with a controller to provide a semi-autonomous system for vehicle safety. We describe how the system is built, present metrics to evaluate its utility and conduct real-time experiments with a safety controller to verify the utility of incorporating the driver's behavior while designing a semi-autonomous system.

To my parents.

Contents

List of Figures

List of Tables

Glossary

α	A probability value between 0 and 1.
$\mathcal{C}(k, \mathcal{I})$	Set of unsafe positions at time k.
Δ	Vehicle prediction multifunction.
u	The control input to the vehicle incorporating the driver and controller input.
δu	The minimum norm input to the vehicle from the controller.
Δu	The change rate of the minimum norm input in the MPC optimization.
g	Vehicle Intervention Function.
$\mathcal{I}$	Current observations of the vehicle and driver.
λ	The semi-autonomous controller function.
$\overline{N}$	The future time horizon.
$\underline{N}$	The past time horizon.
$\mathcal{O}$	The set of prior observations (training data).
o_i	Observation of the vehicle and driver state at time i.
$\mathcal{O}_{\mathcal{M}}$	Prior observations (training data) that belong to mode M.
Ω	A set in $\mathbb{R}^n$.

$\phi_{\overline{N}}(x(k), k)$	Functional description of the environment at position $x(k)$ at time k (Negative is safe, positive is an obstacle).
$\mathcal{R}(k, \mathcal{I})$	Set of reachable states starting at time step k beginning at $x_{i,\overline{N}}$.
$\theta_{\underline{N}}$	The 3D positions of the upper body of the driver from 0 to $\underline{N}$.
$\theta_{i,\underline{N}}$	The 3D positions of the upper body of the driver starting at time i to $i + \underline{N}$.
ε	The discretized state in the MPC optimization.
$x(k)$	State of the vehicle at time k.
$x_{i,N}(k)$	State of the vehicle at time k starting at time index i to $i + N$.
$x(0)$	State of the vehicle at the current time.

Acknowledgements

I would like to thank my advisor Professor Ruzena Bajcsy who has been my advisor since my undergraduate years here at Berkeley and Ram Vasudevan for their guidance throughout this project. I'm grateful for those in the Model Predictive Control Lab: Yiqi Gao, Theresa Lin, Andrew Gray and Professor Francesco Borrelli who I've collaborated with since the beginning of my graduate program and have helped tremendously with this project, and Katherine Driggs-Campbell, Robert Matthew and Aaron Bestick for their help on this and various other projects.

This material is based upon work supported by the National Science Foundation under Awards ECCS-0931437 and ECCS-1239323. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

1 Introduction

Despite the improvement of vehicle safety features in recent years, the number of traffic crashes and fatalities remains high [4]. The National Highway Transportation Safety Administration reported that in 2011, 3,331 people were killed in distracted driving related incidents and another 387,000 were injured. Though it is difficult to quantify why these accidents occur, it is agreed that in-vehicle information systems (e.g. vehicle navigation systems) and other devices such as cell phones make the driver prone to distraction which degrades driving performance. Using data from vehicles instrumented with cameras, several studies have estimated that driver distraction contributes to between 22 to 50% of all crashes [23, 30]. Alarmingly, even with the advances in technology, a recent report conducted indicates that the percentage of fatal crashes due to distraction is rising [6]. In 2011, the Transport Research Laboratory in UK found that distracted drivers spent between 40-60% of the time looking at their phone, resulting in the inability to maintain central lane position and lower response/reaction times [7]. With newer technologies such as augmented reality glasses coming to the market, there are growing concerns about how this will affect vehicle safety.

One potential resolution to the driver distraction problem is the autonomous vehicle [24, 33, 35, 36]. Autonomous vehicles date back to the 1980s starting with Ernst Dickmann at the Bundeswehr University Munich which drove at 63 km/h on streets without traffic [13, 14]. Also in the 1980s, DARPA funded the Autonomous Land Vehicle project bringing together several universities to use laser radar, computer vision and autonomous control to drive at 31 km/h [32, 39]. In the 1990s, the California Partners for Advanced Transportation Technology (PATH) demonstrated a concept called platooning and showed remarkable results with stability and safety [17]. However, due to the advanced infrastructure required, the technology was never realized in industry until the DARPA Grand Challenge 2004, 2005, and 2007, this challenged universities to develop a fully autonomous vehicle on modern infrastructure. Afterwards, Google continued to develop the autonomous vehicle until now, logging hundred of thousands of hours on the road and garnering much attention from the public.

Although autonomous vehicles have taken the public and lawmaker's attention recently, the deployment of these systems has been hindered principally due to the lack of formal methods to verify the safety of such systems in arbitrary situations. Recognizing this shortcoming of theoret-

ical verification algorithms in the face of a growing number of accidents due to driver distraction, several car companies have moved quickly to design active safety systems to avoid accidents. Volvo, for example, has developed one of the most advanced such systems called City Safety that detects potential collisions by treating the driver's input as a disturbance, computing the reachable set of the car over a short time horizon (approximately 500 milliseconds), and immediately decelerating the car if it detects an intersection of the reachable set with an obstacle [11]. Confidence in this autonomous breaking maneuver was achieved via extensive testing over several million kilometers. BMW, Lexus, Hyundai, Ford, and many other car companies have recently entered the semi-autonomous vehicle market. At CES 2013, Toyota presented its semi-autonomous Lexus Advanced Active Safety Research Vehicle which is intended to take-over when it detects an imminent threat [19]. However, these car safety systems have been limited to low speeds since the reachable set at higher speeds grows large.

As these types of semi-autonomous systems become more elaborate and are applied in higher speed situations, threat assessment and driver prediction must occur over a much longer time horizon, which means that the reachable set becomes large. In these situations, we move from a scenario where the driver can simply be treated as a disturbance to a scenario where the driver behavior becomes fundamental to the design of a provably safe semi-autonomous vehicle otherwise the semi-autonomous system begins to behave as a fully autonomous system. However, as each driver may behave differently, a single universal driver model for all semi-autonomous vehicle systems is insufficient. Instead, we must develop a system framework that adjusts to different drivers by taking advantage of real-time empirical observations of the driver such as posture, hand position, gaze, heart-rate and other sensory data. However, once this shift occurs, empirical validation becomes difficult due to the reliance of the system on the driver model itself. Fortunately recent advances in numerical optimization techniques have shown that provably safe controllers can be designed robustly given a particular circumstance [5, 9]. We leverage the framework described in [37] to design a real-time system that can utilize empirical observations of the driver to keep the vehicle safe.

Modeling driver behavior has been studied for decades in a variety of communities. The human factors community, as described in the surveys found in [8, 28], has focused on quantifying the amount of attention required to accomplish a variety of tasks, and the level of distraction due to a variety of common preoccupations that arise during driving. These studies do not provide a

method to predict the driver's behavior given a particular level of distraction and a particular task complexity.

Other communities have used body sensors on the driver [29], eye trackers [15,38], inside cameras [34], or facial tracking [10, 16, 22, 25, 26] to predict the driver's behavior. Although this methods can predict driver intent with 70-90% accuracy with varying prediction times, these approaches do not provide a means to employ the prediction during controller design or a method to quantify the utility of the given prediction in preventing accidents; Trivedi et al at UC San Diego has done tremendous work in the computer vision area in predicting intent and driver behavior and a review of the literature can be found in [16]. For this project, we quantify driver distraction, predict future vehicle states and trajectories, develop statistical models of driving behavior and predictions, and couple this prediction with controller design to construct a provably safe semi-autonomous vehicle framework and illustrate its utility in preventing accidents during an experiment.

We motivate the need for semi-autonomy by considering the "simplest" semi-autonomous vehicle architecture which treats the driver's input into the vehicle as a disturbance. The simplest semi-autonomous vehicle architecture would construct a reachable set for the vehicle and would apply a controller whenever the reach set of the vehicle intersected with an obstacle. Unfortunately this framework is too conservative since by treating the driver's steering, throttle and braking as a disturbance the simplest architecture predicts a large set of potential future trajectories and therefore acts too early and too often. To illustrate this shortcoming, suppose, for example, that the semi-autonomous controller employs a breaking maneuver and does not want to decelerate the vehicle too quickly. At highway speeds, the set of reachable states over a 2 second time horizon is large and likely intersects with some obstacle all the time. As a result, the controller would continually believe that the driver was in trouble and constantly take over control of the vehicle, even though the driver most likely would be safe.

Remark 1: *We are ultimately interested in the control of a vehicle whose evolution is given as the solution of the* vehicle dynamics*:*

$$x(k+1) = f(x(k), u(k)), \quad \forall k \in \mathbb{N}, \tag{1}$$

where $x(k) \in \mathbb{R}^n$ is the state of the vehicle at time k, $u(k) \in U$ is the input into the vehicle dynamics at time k where $U \subset \mathbb{R}^m$ is a compact, connected set containing the origin, and $x(0)$, the initial state, is assumed given. For a fixed future time horizon $\overline{N} \in \mathbb{N}$ the constraint function*,*

$\phi_{\overline{N}} : \mathbb{R}^n \times \{0, \ldots, \overline{N}\} \to \mathbb{R}$*, describes the safe part of the environment which the vehicle travels in. That is, the state of the car remains outside of obstacles in the environment (safe) at a time* k *if* $\phi_{\overline{N}}(x(k), k) \leq 0$*, by definition. Observe that if we know the location of the obstacles at* $k = 0$ *but are only aware of bounds on the movement of each obstacle, then we can still encapsulate these bounds into the constraint function,* $\phi_{\overline{N}}$*.*

An outline of this paper is as follows: in Section 2, we present a real-time implementation of the framework described in [37] that uses a driver model to predict likely vehicle behavior by using empirical observations and another component that determines when to intervene. In Section 2.2, we present several metrics to evaluate the performance of the driver model and real-time system. Section 3 describes the implementation of the vehicle model, controller, and simulator. In Section 4, we construct a real-time driver-in-the-loop experiment and illustrate the utility of incorporating empirical observations of the drivers poser in the control loop to prevent accidents. Section 5 describes how the data is processed. Section 6 provides an evaluation of the system with respect to the metrics and Section 7 concludes this manuscript.

2 Semi-Autonomous Vehicle Framework

2.1 Problem Formulation

Our semi-autonomous vehicle framework divides the problem of determining when to intervene into three components: driver prediction, intervention detection, and autonomous control.

The first component of the proposed three component semi-autonomous vehicle framework predicts driver behavior and is called the Vehicle Prediction Multifunction (VPM). A high level explanation is as follows: given prior knowledge of driving behavior and the current state of the vehicle and driver, the VPM predicts a set of likely future trajectories and inputs that the driver will take in the next $\overline{N}$ time steps. Formally, we incorporate the prior observations denoted $\mathcal{O}$ (this may include past states of the driver, past inputs, past vehicle trajectories, and past environment scenarios from training data) and the current information $\mathcal{I} = \{\overline{N}, \underline{N}, x(0), \phi_{\overline{N}}, \theta_{\underline{N}}\}$ (where $\underline{N} \in \mathbb{N}$ denotes the past time horizon and $\theta_{\underline{N}}$ represents the observations of the human for the last $\underline{N}$ time steps) in order to predict a smaller and more useful set of potential states for the vehicle and the future expected steering. The component is defined below:

Component 1: *Vehicle Prediction Multifunction (VPM): Fixing $\overline{N}, \underline{N} \in \mathbb{N}$ (future and past time horizons respectively) and choosing $\alpha \in [0, 1]$ (probability of prediction accuracy) and $k \in \{0, \ldots, \overline{N}\}$, construct the* vehicle prediction multifunction *denoted $\Delta(\alpha, k, \mathcal{O}, \mathcal{I})$ as the solution to:*

$$\begin{aligned} &\underset{\Omega \subset \mathbb{R}^n}{\operatorname{argmin}} \ |\Omega| \\ \text{s.t.} \quad &P\left((X(k) - x(0)) \subset \Omega \mid \mathcal{O}, \mathcal{I}\right) \geq \alpha, \end{aligned} \tag{2}$$

where $|\Omega|$ denotes the area of the set Ω, and $P\left((X(k) - x(0)) \subset \Delta \mid \mathcal{O}, \mathcal{I}\right)$ denotes the probability that the difference between a solution to Equation (1) *beginning at $x(0)$ in k time steps, denoted $X(k)$, and $x(0)$ is inside of the set Ω given the prior observations and prior information (note $X(k)$ is capitalized in order to distinguish it as the random variable).*

The vehicle prediction multifunction is the smallest size set that contains α probable vehicle trajectories in k time steps given prior observations and current information. In order to construct this function, we require a model linking $X(k)$ with prior observations, which we obtain by learning

driver behavior. In essence, $X(k)$ represents likely future trajectories and $\Delta(\alpha, k, \mathcal{O}, \mathcal{I})$ represents the set that encapsulates an α portion of those trajectories. In Section 3.3, we describe how we construct the required conditional distribution and how we construct the vehicle prediction multifunction. .

Using the vehicle prediction multifunction and the constraint function, the second component of the semi-autonomous vehicle framework is a binary-valued function that determines when the driver is in need of assistance. This is defined formally as follows:

Component 2: *The* vehicle intervention function *(VIF) denoted g, is 1 when the driver requires aid and is defined as:*

$$g(\alpha, \mathcal{O}, \mathcal{I}) = \begin{cases} 1 & \textit{if } \cup_{k=0}^{\overline{N}} \left(f(\Delta(\alpha, k, \mathcal{O}, \mathcal{I})) \cap \mathcal{C}(k, \mathcal{I}) \right) \neq \emptyset \\ 0 & \textit{otherwise,} \end{cases} \tag{3}$$

where $\mathcal{C}(k, \mathcal{I}) = \{z \in \mathbb{R}^n \mid \phi_{\underline{N}}(z - x(0), k) > 0\}$, the set of unsafe positions, and $f : \Delta \to \mathbb{R}$, is a function that operates on the set from $\Delta(\alpha, k, \mathcal{O}, \mathcal{I})$.

The vehicle intervention function is 1 if a function f of α probable vehicle trajectory intersects an obstacle in the next $\overline{N}$ time steps given our past observations and current information. In our previous work [37], the function f used was the identity relation. For this project, we consider the average of the set, which corresponds to the driver's expected future trajectory.

If the VIF determines that the driver needs aid, then $f(\Delta(\alpha, k, \mathcal{O}, \mathcal{I}))$ is sent to the vehicle's semi-autonomous controller to determine a necessary minimum-norm input to augment the expected driver's input to ensure that the vehicle stays within the safety constraints.

Component 3: *The controller, denoted $\lambda : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{\overline{N}}$, determines an input to the vehicle for the next $\overline{N}$ time steps and is defined as:*

$$\delta u = \lambda(f(\Delta(\alpha, k, \mathcal{O}, \mathcal{I})), \mathcal{C}(k, \mathcal{I})) \tag{4}$$

where δu can be either an augmented input to the driver's steering or the actual steering for fully autonomous control.

Algorithm 1 describes our semi-autonomous vehicle framework.

Algorithm 1 Semi-Autonomous Vehicle Framework

1: Data: $\overline{N}, \underline{N} \in \mathbb{N}$ and $x(0) \in \mathbb{R}^n$
2: Let $u(k) = 0, \forall k \in \{0, \ldots, \overline{N}\}$.
3: **for** Each time step **do**
4: update $\phi_{\overline{N}}$ and $\theta_{\overline{N}}$, and set $x(0)$ equal to the vehicle's current state.
5: **if** $g(\alpha, \mathcal{O}, \mathcal{I}) == 1$ **then**
6: set $\delta u = \lambda(f(\Delta(\alpha, k, \mathcal{O}, \mathcal{I})), \mathcal{C}(k, \mathcal{I}))$
7: apply δu to the vehicle
8: **end if**
9: **end for**

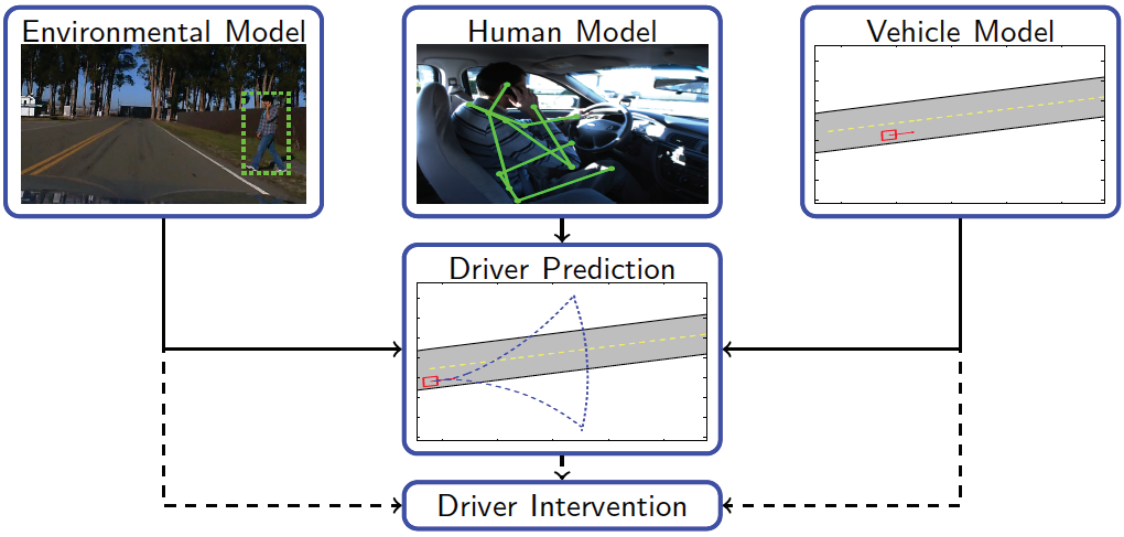

Figure 1: Semi-autonomous framework with driver modeling.

In the algorithm, line 4 updates the environmental constraints and observations of the driver; line 5 represents the calculation of the vehicle prediction multifunction and intervention function; line 6-7 represents the application of the semi-autonomous controller.

The idea behind this framework is to modularly incorporate driver, vehicle, and environment data. The driver model provides information on the state of the driver in real time and predicts likely future driving behavior, in either steering wheel angle space or real world space. It is further described in Section 3.3. The environment model provides information on outside obstacles, pedestrians, and road curvature. It is further described in Section 3.1. The vehicle model is used in the vehicle safety controller which incorporates the environment model to determine an input to

keep the car safe if necessary. It is described in Section 3.2.

2.2 Metrics to Evaluate Performance

To evaluate the performance of our vehicle prediction multifunction and the vehicle intervention function, we utilize the 4 metrics presented in [37] described below.

In this subsection, we define four separate metrics: the *Accuracy of the Vehicle Prediction Multifunction* (AVPM), the *Precision of the Vehicle Prediction Multifunction* (PVPM), the *Recall of the Vehicle Intervention Function* (RVIF), and the *Precision of the Vehicle Intervention Function* (PVIF). The AVPM and the PVPM measure the informativeness of our vehicle prediction multifunction and the RVIF and the PVIF measure the utility of the vehicle intervention function.

While the most informative set prescribed by the vehicle prediction multifunction would predict a single vehicle trajectory that was always accurate, this requires perfect knowledge of the intent of the driver. Instead, we forecast a set of potential trajectories. Intuitively, AVPM denotes how accurate the set is and PVPM denotes how much smaller the predicted set of trajectories is compared to the reachable set. Notice that the AVPM can act either on steering wheel angles or the vehicle trajectories while the PVPM only can act on the vehicle trajectories. During our experiment in Section 4, we measure the informativeness of our prediction by first determining if the observed vehicle behavior for the next $\overline{N}$ time steps was inside of the predicted set at each of the next $\overline{N}$ time steps. Explicitly, fixing $\alpha \in [0,1]$ suppose we are given M distinct observed vehicle trajectories for $\overline{N}$ time steps beginning at time step i denoted $\{x_{i,\overline{N}} : \{0,\dots,\overline{N}\} \to \mathbb{R}^n\}_{i=1}^M$ then:

$$\text{AVPM} = \frac{1}{M}\sum_{i=1}^{M}\prod_{k=0}^{\overline{N}} \mathbb{1}\{x_{i,\overline{N}}(k) \in \Delta(\alpha, k, \mathcal{O}, \mathcal{I}_i)\}, \tag{5}$$

where $\mathbb{1}$ is the indicator function and the current information, $\mathcal{I}_i$, is indexed by the time step i since the vehicle prediction multifunction is a function of the initial condition of the vehicle, $x_{i,\overline{N}}(0)$. We also measure the informativeness of the predicted set by computing one minus the area of the predicted set for the next $\overline{N}$ time steps over the area of the set of reachable states for the next $\overline{N}$ time steps:

$$\text{PVPM} = 1 - \frac{1}{M}\sum_{i=1}^{M}\frac{|\cup_{k=0}^{\overline{N}}\Delta(\alpha, k, \mathcal{O}, \mathcal{I}_i)|}{|\cup_{k=0}^{\overline{N}}\mathcal{R}(k, \mathcal{I}_i)|}, \tag{6}$$

where $\mathcal{R}(k, \mathcal{I}_i)$ is the set of reachable states at time step k beginning at $x_{i,\overline{N}}(0)$. We normalize by the set of reachable states since it is always accurate, but imprecise because it assumes that the vehicle behaves arbitrarily.

Similarly, the most useful vehicle intervention function would only act if and when the vehicle was actually going to get into an accident. Unfortunately the determination of this most useful intervention procedure is difficult since it demands being clairvoyant. To understand these concepts in this context, we borrow several definitions from the binary classification literature [12]. We define an observed $\overline{N}$ time step vehicle trajectory to be "true" if an accident is actually observed at any instance in the next $\overline{N}$ time steps, and we define an observed $\overline{N}$ time step vehicle trajectory to be "false" if an accident is not observed in any of the next $\overline{N}$ time steps. We define an observed vehicle trajectory to be "positive" if the vehicle intervention function (i.e. the classifier in our context) returns 1, and we define an observed vehicle trajectory to be "negative" if the vehicle intervention function returns 0.

The RVIF is then defined as the true-positive observations divided by the sum of the true-positive observations and false-negative observations (note that in the binary classification literature this is when the observation is true, but the classifier returns negative) and measures the sensitivity of the vehicle detection function. *If the RVIF is* 1*, then the semi-autonomous vehicle framework intervened whenever an accident occurred*. Notice that if the vehicle intervention function relied on the set of reachable states, then there would be no false-negative observations since an accident could have only occurred if there was a non-trivial intersection between the reachable set and an obstacle. Therefore, this choice of vehicle intervention function would always be perfect with respect to this metric.

The PVIF is defined as the true-positive observations divided by the sum of the true-positive observations and false-positive observations (note that in the binary classification literature this is when the observation is false, but the classifier returns positive). *If the PVIF is much less than* 1 *then the semi-autonomous vehicle framework intervened far more than was actually required*. If this is the case the vehicle begins to behave more like an autonomous vehicle. Suppose again that the vehicle intervention function relied on the set of reachable states, then there would most likely be a large number of false-positive events since the reachable set would intersect with some obstacle in the neighboring lanes that the driver in all likelihood would never have an accident

with. Our goal in the remainder of this paper is to illustrate the informativeness and utility of considering prior observations before the current time step, the driver state function and the vehicle trajectory generated by the numerical optimization algorithm for the current time step in the semi-autonomous vehicle framework by using these four metrics.

3 System Architecture

In this section, we describe in detail each of the components required in Section 2.

3.1 Environment model

The environment model provides information on the state of the road and obstacles such as pedestrians or other vehicles. This can be obtained through commercial off-the-shelf systems such as Mobileye [3] or radar systems now installed in many vehicles. In our experiments described in Section 4, our simulator provides radar with known road curvature so we have complete certainty of the environment.

3.2 Vehicle model

This subsection was written by Yiqi Gao in Professor Francesco Borrelli's MPC lab and included for clarity's sake.

In this section, we present the mathematical models used for control design. Consider the vehicle sketch in Figure 2. We use the following set of differential equations to describe the vehicle motion within the lane,

$$m\ddot{x} = m\dot{y}\dot{\psi} + 2F_{xf} + 2F_{xr}, \tag{7a}$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + 2F_{yf} + 2F_{yr}, \tag{7b}$$

$$I_z\ddot{\psi} = 2aF_{yf} - 2bF_{yr}, \tag{7c}$$

$$\dot{e}_\psi = \dot{\psi} - \dot{\psi}_d, \tag{7d}$$

$$\dot{e}_y = \dot{y}\cos(e_\psi) + \dot{x}\sin(e_\psi), \tag{7e}$$

$$\dot{s} = \dot{x}cos(e_{psi}) - \dot{y}sin(e_\psi), \tag{7f}$$

where m and I_z denote the vehicle mass and yaw inertia, respectively, a and b denote the distances from the vehicle center of gravity to the front and rear axles, respectively. $\dot{x}$ and $\dot{y}$ denote the vehicle longitudinal and lateral velocities, respectively, and $\dot{\psi}$ is the turning rate around a vertical

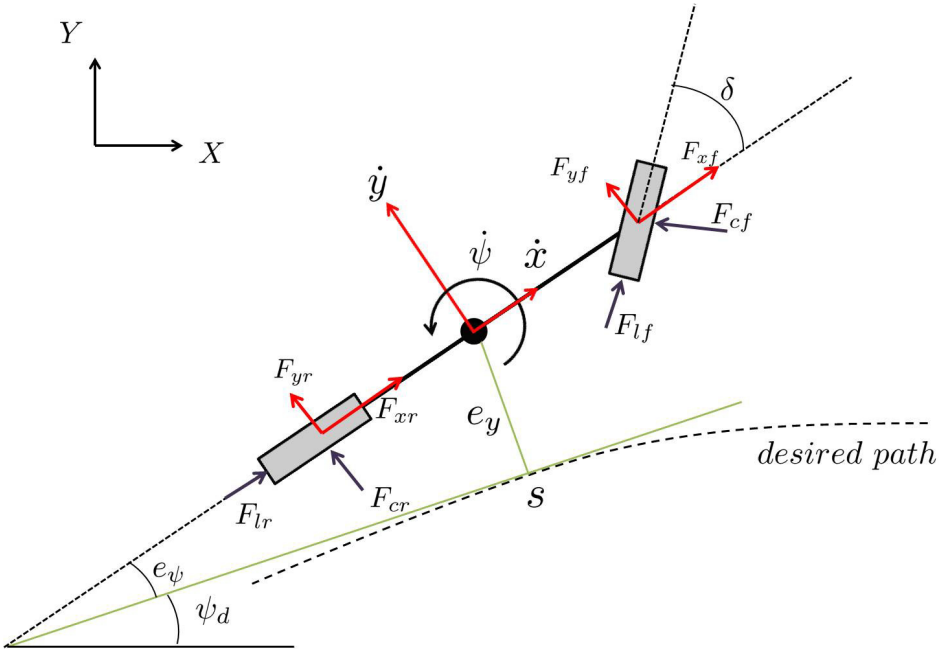

Figure 2: Modeling notation depicting the forces in the vehicle body-fixed frame ($F_{x\star}$ and $F_{y\star}$), the forces in the tire-fixed frame ($F_{l\star}$ and $F_{c\star}$), and the rotational and translational velocities. The relative coordinates e_y and e_ψ are illustrated on the sketch of the road as well as the road tangent ψ_d.

axis at the vehicle's center of gravity. e_ψ and e_y in Figure 2 denote the vehicle orientation and lateral position, respectively, in a road aligned coordinate frame and ψ_d is the angle of the tangent to the road centerline in a fix coordinate frame. s is the vehicle longitudinal position along the lane center.F_{yf} and F_{yr} are front and rear tire forces acting along the vehicle lateral axis, F_{xf} and F_{xr} forces acting along the vehicle longitudinal axis

The longitudinal and lateral tire force components in the vehicle body frame are modeled as,

$$F_{x\star} = F_{l\star}\cos(\delta_\star) - F_{c\star}\sin(\delta_\star), \tag{8a}$$

$$F_{y\star} = F_{l\star}\sin(\delta_\star) + F_{c\star}\cos(\delta_\star), \tag{8b}$$

where $\star$ denotes either f or r for front and rear tire, $\delta_\star$ is the steering angle at wheel. We introduce the following assumption on the steering angles.

Assumption 1: *Only the steering angle at the front wheels can be controlled. i.e., $\delta_f = \delta$ and $\delta_r = 0$. In addition, an actuator which corrects the driver commanded steering angle, such that $\delta = \delta_d + \delta_c$, is available, where δ_d is the driver commanded steering angle and δ_c is the correcting*

steering angle component.

The longitudinal force in the tire frame, $f_{l\star}$ is calculated from the equation

$$f_{l\star} = \beta_r \mu F_{z\star} \tag{9}$$

where $\beta_r \in [-1, 1]$ is referred to as the braking ratio. $\beta_r = -1$ corresponds to full braking and $\beta_r = 1$ corresponds to full throttle. $f_{c\star}$ is computed using a modified nonlinear Fiala tire model [21]:

$$f_{c\star} = \begin{cases} -C_\alpha tan(\alpha_\star) + \frac{C_\alpha^2}{2\eta\mu F_{z\star}} |tan(\alpha_\star)| tan(\alpha_\star) \\ \quad - \frac{C_\alpha^3}{27\eta^2\mu^2 F_{z\star}} tan^3(\alpha_\star), \ \textbf{if} \ |\alpha_\star| < \alpha_{sl} \\ -\eta\mu F_{z\star} sgn(\alpha_\star), \qquad \textbf{if} \ |\alpha_\star| \geq \alpha_{sl} \end{cases} \tag{10}$$

where $\alpha_\star$ denotes the tire slip angle, μ denotes the friction coefficient, and $Fz_\star$ denotes the vertical load at each wheel. C_α is the tire cornering stiffness and $\eta = \frac{\sqrt{\mu^2 F_z^2 - f_l^2}}{\mu F_z}$ which can be written as $\eta = \sqrt{1 - \beta_r^2}$. $\alpha_{sl} = tan^{-1}(\frac{3\eta\mu F_z}{C_\alpha})$ denotes the saturation point of the slip ratio. μ is assumed to be a known constant and is the same for all wheels. We also assume the vertical forces $F_{z\star}$ to be constants and are determined by the steady state weight distribution of the vehicle with no accelerations at the center of the gravity.

The slip ratios $\alpha_\star$ are approximated as follows:

$$\alpha_f = \frac{\dot{y} + a\dot{\psi}}{\dot{x}} - \delta, \ \ \alpha_r = \frac{\dot{y} - b\dot{\psi}}{\dot{x}} \tag{11}$$

We write the model (7) to (11) in the following compact form:

$$\dot{\xi}(t) = f(\xi(t), u(t)) \tag{12}$$

where $\xi = [\dot{x}, \dot{y}, \dot{\psi}, e_\psi, e_y, s]$ and $u = [\delta, \beta_r]$ are the state and input vectors respectively.

3.3 Component 1: Vehicle Prediction Multifunction (VPM)

In this subsection, we describe how we construct component 1, the vehicle prediction multifunction based off of work in [37]. During the experiment considered in Section 4, we want to illustrate the utility of considering various types of prior observations and current information. Therefore, in this section we consider the construction of the required distribution in generality.

We hypothesize that given a level of distraction for the driver and a specific environment, the distribution of the potential vehicle trajectories at a particular time step has distinct modes. For example, the distribution of potential vehicle trajectories when the driver is attentive will be different for when the driver is distracted; likewise, the vehicle trajectories for a curved and straight road will differ. To recover these distinct modes, we apply a k-means clustering algorithm on the prior observations, $\mathcal{O}$, and present two types of probability distributions for each mode: one using the empirical cumulative distribution defined on the observations within that mode [20, 31] and one by fitting the potential vehicle trajectories to a gaussian at each particular time step. For each mode $\mathcal{M}$, there is a set $\mathcal{O}_{\mathcal{M}} \subset \mathcal{O}$ of observations associated with each mode. Notice that we are currently performing unsupervised learning.

3.3.1 Empirical distribution VPM

For each mode $\mathcal{M}$, we have $\mathcal{O}_{\mathcal{M}} = \{o_i\}_i$ where i denotes the time step and each o_i is a vector composed of observations (steering angle, environment constraints and postures) for that mode. Fixing $\overline{N} \in \mathbb{N}$ with respect to each observation o_i, we can associate the actual observed vehicle input and trajectory for the next $\overline{N}$ time steps beginning at time step i which we denote by $u_{i,\overline{N}} : \{0, \ldots, \overline{N}\} \to \mathbb{R}$ and $x_{i,\overline{N}} : \{0, \ldots, \overline{N}\} \to \mathbb{R}^n$.

As Algorithm 1 runs at each time step, we are given current information, $\mathcal{I}$, which includes the vehicles current initial condition which we denote by $x(0)$, determine which mode the driver is in, and outputs the set $\Delta(\alpha, k, \mathcal{O}_{\mathcal{M}}, \mathcal{I})$.

Letting $\mathcal{M}$ denote the mode to which $\mathcal{I}$ belongs, for some $z_1, z_2 \in \mathbb{R}^n$, let $\Omega = \{x \in \mathbb{R}^n | z_1 \leq x \leq z_2\}$ and we define the probability that the future steering at time step k, denoted $x(k)$, and the

vehicle's initial condition $x(0)$ is in Ω (between z_1 and z_2) as:

$$P(z_1 \leq x(k) - x(0) \leq z_2 \mid \mathcal{O}, \mathcal{I}) = \frac{1}{|\mathcal{M}|} \sum_{\forall i \in \mathcal{M}} \mathbb{1}\{z_1(k) \leq (x_{i,\overline{N}}(k) - x_{i,\overline{N}}(0)) \leq z_2(k)\} \quad (13)$$

where $|\mathcal{M}|$ denotes the number of elements in mode $\mathcal{M}$. During our experiment in Section 4, we find the minimum z_1 (most counter-clockwise turn) and maximum z_2 (most clockwise turn) such that the probability in Equation (13) is 1.

Therefore, recalling Δ in Section 2, we define $\Delta(\alpha, k, \mathcal{O}_{\mathcal{M}}, \mathcal{I})$ as:

$$\begin{aligned} &\underset{\Omega \subset \mathbb{R}^n}{\operatorname{argmin}} \ |\Omega| \\ \textit{s.t.} \quad &\frac{1}{|\mathcal{M}|} \sum_{\forall i \in \mathcal{M}} \mathbb{1}\{z_1(k) \leq (x_{i,\overline{N}}(k) - x_{i,\overline{N}}(0)) \leq z_2(k)\} = 1, \end{aligned} \quad (14)$$

3.3.2 Gaussian distribution VPM

To apply the VPM to a stochastic controller, we hypothesize that in a given scenario, the driver's future trajectory and steering wheel angles can be described by a gaussian distribution over time. To construct the gaussian distribution, we construct P in the following manner: Let $x_m(k)$ be the average steering corresponding to associated observed vehicle behavior and $\sigma_m(k)$ corresponding to the standard deviation of steering behavior in mode $\mathcal{M}$.

Letting $\mathcal{M}$ denote the mode to which $\mathcal{I}$ belongs, for some $z \in \mathbb{R}^n$ we define the probability that the future steering at time step k, denoted $x(k)$as:

$$P((x(k) - x(0)) \leq z \mid \mathcal{O}, \mathcal{I}) = \frac{1}{\sigma_m(k)\sqrt{2\pi}} e^{-\frac{(x(k) - x_m(k))^2}{2\sigma_m^2(k)}} \quad (15)$$

3.4 Component 2 and 3: Robust Predictive Control Problem

Once component 1 and 2 predict that the driver may run into trouble, our controller will apply an input to the system to keep the vehicle safe. This problem of lane keeping and obstacle avoidance can be formulated as a Model Predictive Control (MPC) Problem. At each sampling time, an finite

time constrained optimal control problem is solved to determine an optimal input sequence. The computed input is only applied to the system for a finite time interval before the MPC problem is re-solved using new measurements from sensors to find the new input.

The general MPC optimization problem can be written as:

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \; & J(x(t), \bar{u}) \\ s.t. \; & x_{k+1} = f(x_k, u_k), k = 0, \ldots, \overline{N} - 1 \\ & h(x_k, u_k) \leq 0, k = 0, \ldots, \overline{N} - 1 \\ & x_{\overline{N}} \in \chi_f \\ & x_0 = x(0) \end{aligned} \tag{16}$$

where $\bar{u}$ is the optimal input, x is the state, f is discrete dynamical equation, h are the constraints (where a negative value means the constraint is satisfied), χ_f is the target set, and x_0 is the initial state.

First, we looked at 2 types of control inputs to put into the system. The first method, which we describe and use in [37], is where the controller fully takes over when the driver model predicts that the driver may be in danger. The second method, which we use, uses the expected driver steering as the initial input and tries to apply the minimum norm input (aka least input required) to keep the vehicle safe.

This method is formulated below:

We discretize the system (12) with a fixed sampling time T_s to obtain,

$$\xi_{k+1} = f_{\mathrm{d}}(\xi_k, u_k), \tag{17}$$

and formulate the optimization problem being solved at each time instant as

$$\min_{\bar{\mathbf{u}},\varepsilon} \sum_{k=0}^{\overline{N}-1} ||\delta u_{t+k,t}||_R^2 + ||\Delta u_{t+k,t}||_S^2 + \lambda\varepsilon \tag{18a}$$

$$s.t.\ \xi_{t+k+1,t} = f_{\mathrm{d}}(\xi_{t+k,t}, u_{t+k,t}), \tag{18b}$$

$$u_{t+k,t} = \delta u_{t+k,t} + u_{t+k,t}^{driver}, \tag{18c}$$

$$\delta u_{t+k,t} = \delta u_{t+k-1,t} + \Delta u_{t+k,t}, \tag{18d}$$

$$h(\xi_{t+k,t}, u_{t+k,t}) \leq \mathbf{1}\varepsilon,\ \ \varepsilon \geq 0, \tag{18e}$$

$$u_{t+k,t} \in \mathcal{U}, \tag{18f}$$

$$\Delta u_{t+k,t} \in \Delta\mathcal{U}, \tag{18g}$$

$$\xi_{t,t} = \xi(t), \tag{18h}$$

where t denotes the current time instant and $\xi_{t+k,t}$ denotes the predicted state at time $t+k$ obtained by applying the control sequence $\mathbf{u} = \{u_{t,t}, \ldots, u_{t+k,t}\}$ to the system (17) with $\xi_{t,t} = \xi(t)$. (18e) denotes the state constraints imposed by lane keeping and obstacle avoiding and have been imposed as soft constraints, by introducing the slack variable ε in (18a) and (18e). A more detailed description on the constraint formulation can be found in [18]. $u_{t+k,t}^{driver}$ is the predicted driver steering input at time step k, provided by the driver model. $\delta u_{t+k,t}$ is the correcting control input to the driver's input. $\Delta u_{t+k,t}$ is the change rate of the correcting control to the driver's input. R, S and λ are weights of appropriate dimension penalizing the control correcting, change rate of control correcting, and violation of the soft constraints, respectively.

This MPC problem is solved every 200ms whether or not the driver model predicts that the driver will be safe or not. When the driver is expected to be safe, it is trivial to see that δu is zero at optimal. However, when the driver is expected to perform unsafely, the MPC problem will supply some input to keep the vehicle safe.

4 Experimental Design

The follow section describes the experiment that was conducted to analyze the proposed framework. A large part of this experimental design was inspired by the survey [7]. The purpose of these experiments was to show that the predicted future set of the driver model is informative with more information. A study by Redelmeier and Tibshirani [27] found little correlation in accident risk between those having hand-held phones and hands-free mobile devices, suggesting that risk of collision is not related to physically holding the phone. Therefore, to test the effect of distraction or quantify distraction, we had to distract the driver by having them send a text message on the phone.

4.1 Experimental Setup

The workflow, illustrated in Figure 3 and experimental setup, as illustrated in Figure 4 and 5 consists of a driving simulator, a Microsoft Kinect, and smartphone to simulate distraction. We use a high end gaming steering wheel and pedals (Logitech G-25) which interfaces with the driving simulator through Simulink. The driving simulator, CarSim 8.02, communicates with a real-time dSpace DS1006 hardware-in-the-loop (HIL) over a fiber-optic link for the physics and model computations at 50 frames per second and a real-time dSpace DS1401 MicroAutoBox over an ethernet link for the MPC controller [$1, 2$] . CarSim's vehicle model is proprietary, but CarSim outputs information at each time step about the state of the vehicle and environment. This includes information at each time step about the global coordinates of the car, the path of the center of the road in global coordinates, the radius of curvature of the road, the longitudinal and lateral velocities of the car at each time step, the location of all of the obstacles in global coordinates, the longitudinal and lateral velocities of all of the obstacles, and a radar sensor telling us which obstacles are observable from the vehicle.

The dSpace DS1006 box sends the vehicle state via CANbus to the dSpace DS1401 MicroAutoBox which runs the MPC controller. CANbus, short for Controller Area Network bus, is the industry standard that allows microcontrollers and devices to communicate with each other in vehicles. The MPC optimization problem is solved every 200ms and sends the controlled vehicle

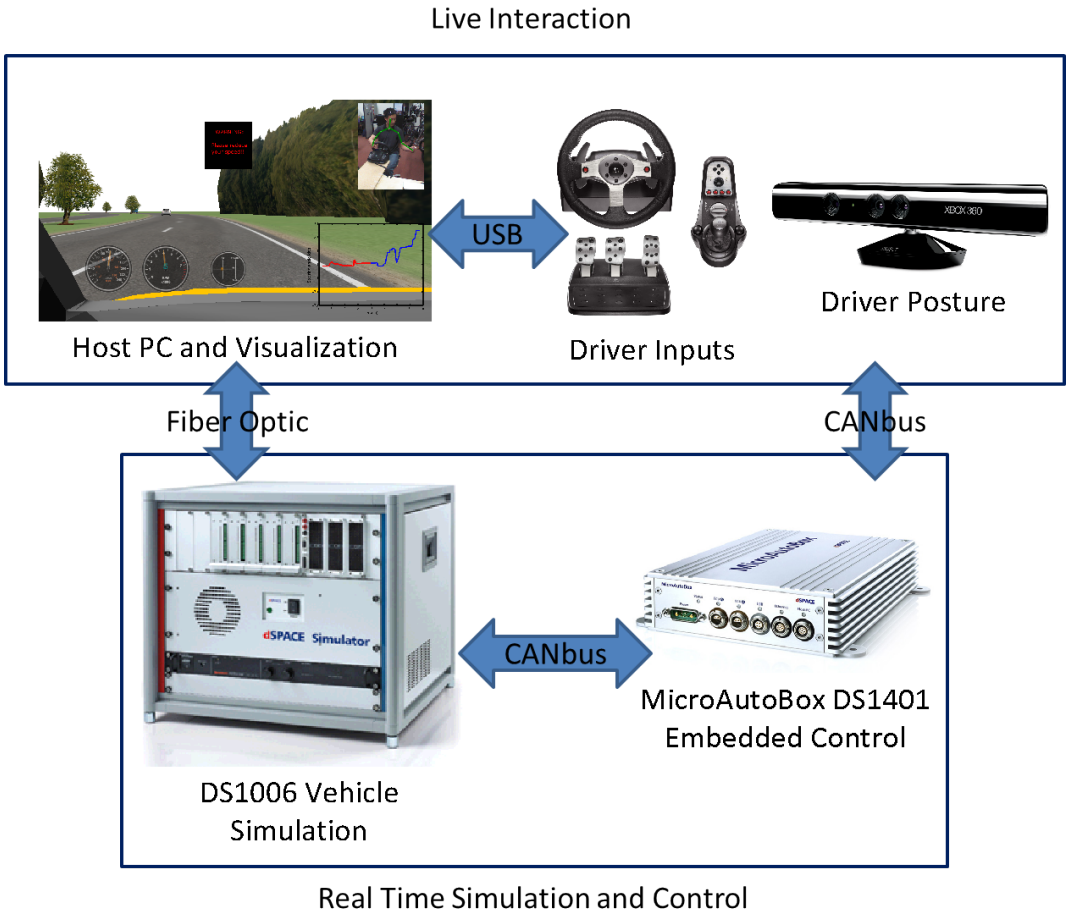

Figure 3: Schematic of the vehicle simulator

state via CANbus back to the DS1006 box. The DS1401 box also sends data via an ethernet link to the main driving simulator machine for recording.

The Microsoft Kinect is used to obtain joint locations in 3D space for the upper extremities at 30 Hz. The DS1401 box sends the vehicle state via CANbus to the driver model running on the main computer, which predicts future driving behavior in under 10ms, and returns it to the DS1401 for the MPC optimization procedure.

To keep the text messages consistent, we provide an android smartphone for the subject to use. A simple app used in [37] was used and displayed multiple questions. The response was not stored or checked.

Each system is time synchronized via an NTP server. Each system also has different sampling

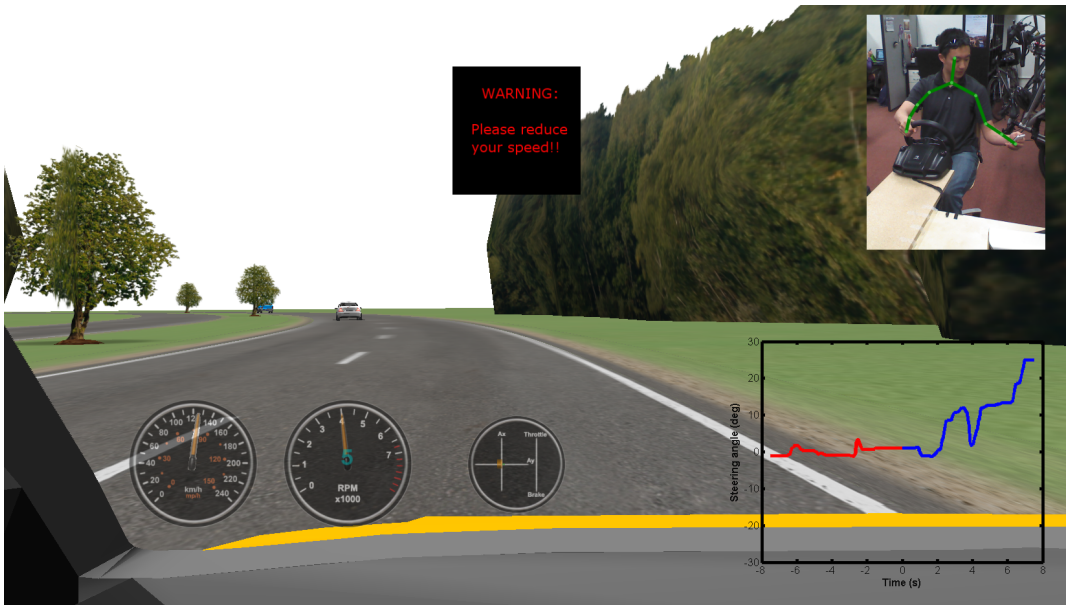

Figure 4: Simulated environment from the driver's view used in our experiment with posture data from the Kinect and steering data from dSpace. Top-right: Human skeleton markers resulting from the Kinect data. Bottom-left: Steering angle for past, current and future.

Figure 5: An example of an experimental setup. During the experiment, the subject was prevented from viewing the left monitor.

Mode	Sampling rate
Kinect	$30fps$
DS1006	$100Hz$
DS1401	$5Hz$
Car Simulator	$50Hz$

Table 2: Sampling rates of the different systems

rates listed in Table 2.

The experiment was conducted over one 2 hour session. The first hour was for data collection and model learning. The second hour for model validation. The subject was asked to keep the vehicle's speed between 65 and 70 miles per hour and to drive safely. The subject was asked to stay in the lane unless an emergency forces him to depart from the lane. There was traffic in the opposite lane and also in the driver's lane. The traffic in the same lane (except during unexpected events) travels at the same longitudinal velocity as the subject's vehicle at any time, so the subject can never reach those vehicles. This was done to limit the variability of potential interactions. In order to simulate learning of the roadway, which is common for day-to-day commutes, we used the same courses for a few of the same training and testing sets.

4.2 Generating the driver model

The driver model was trained on four 10 minute courses. Scenario 1 consists of a two-way, one lane roadway with many turns and a few known objects in the way. Scenario 2 consists of a course with 22 left and right turns. Scenarios 3 consists of a 2-way, 1 lane roadway. Scenario 4 consists of a straight 2 lane highway with occasional turns. In all scenarios, the driver occasionally had to respond to text messages when prompted.

4.2.1 Training Scenario 1

Eights times throughout the drive, the driver received a text message and responded to it accordingly. This required the driver to reach for the phone, look down to read the message, respond and put the phone back.

4.2.2 Training Scenario 2

There were 14 left/right turns. Half of the turns were taken when the driver was texting.

4.2.3 Training Scenario 3

Eights times throughout the drive, the driver responded to text messages. Three minutes into the drive, the driver experienced minor traffic. Five minutes into the drive, a lead car appeared. Seven minutes into the drive, the lead car began to slow down while the driver was responding to the text message.

4.2.4 Training Scenario 4

The driver was asked to stay in the right most lane, following a lead car, as much as possible. Every 1200 meters, the lead car slowed down to a complete stop requiring the driver to make a lane change. Half of the time, the user was asked to check a text message when the lead car began to slow down.

4.3 Model verification

Based on our previous paper, we tested the driver model that utilized the driver posture for two seconds and four seconds of future environment for our real-time implementation [37]. The driver model was tested on four 10 minute courses. To test the efficacy of the controller, the controller

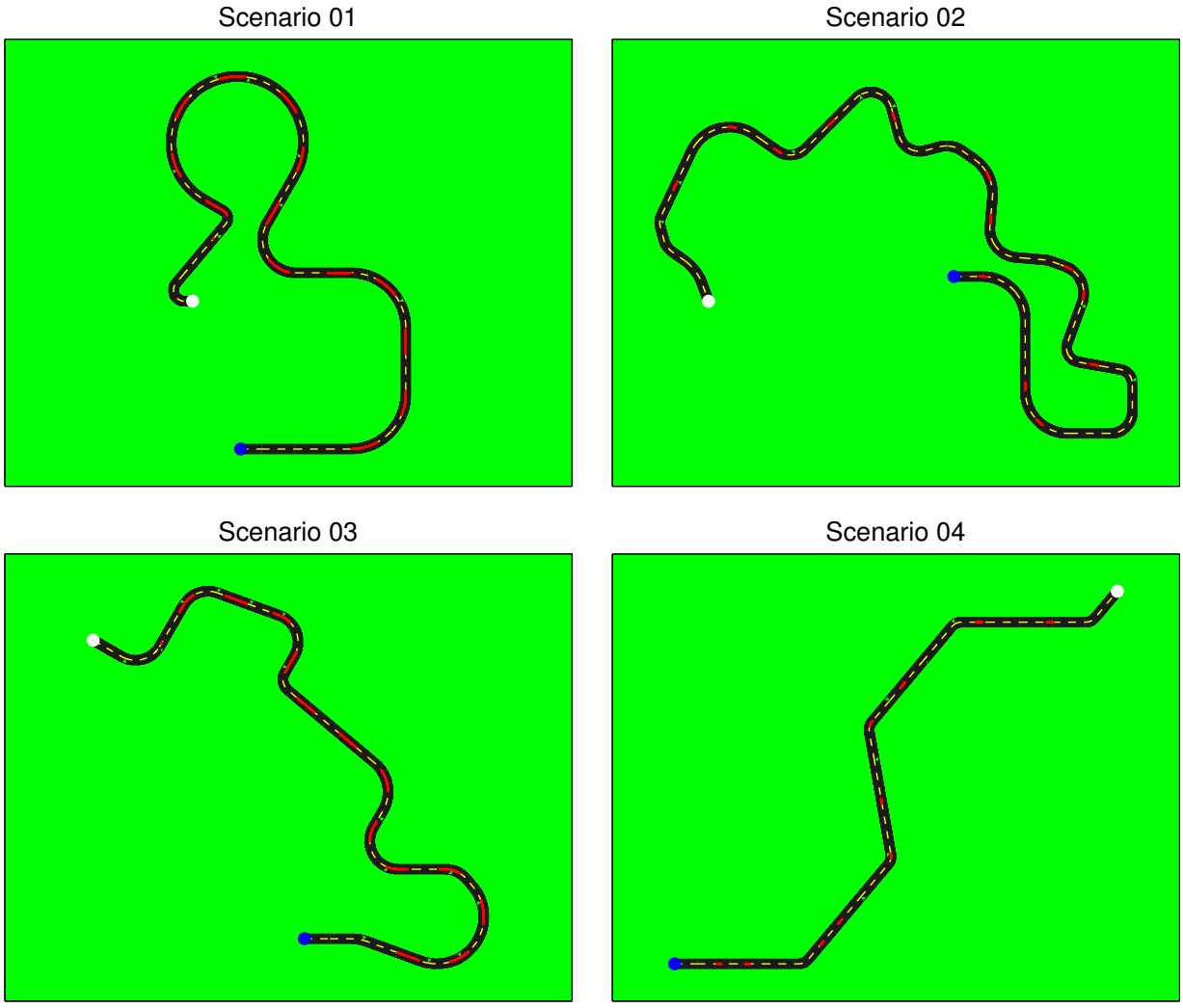

Figure 6: The various training and testing scenarios. The blue dot represents the starting point, white dot represents the end point, and red lines when the driver was asked to answer a text message.

was active for the first three scenarios. To test the other models, we turned off the controller for the last scenario.

5 Data Processing

This section describes how the raw data is processed to create the driver model.

5.1 Steering and Vehicle Data

The steering and other vehicle data is obtained via 2 methods: through the CarSim saved file and through the CANbus from DS1006. We record absolute steering angle from 0 degrees which corresponds to straight driving. The CarSim data outputs steering angles at 100Hz and CANbus outputs steering angles at 20Hz. A positive steering angle corresponds to a left turn and negative steering angle corresponds to a right turn.

We also record the yaw and e_y (error of the vehicle from the centerline). A positive yaw angle corresponds to a left heading with respect to the centerline of the road. A positive e_y means the car is to the left of the centerline.

For our driver model, the steering data is fed to the clustering algorithm without any modification as a stacked vector.

5.2 Environmental Data

The environmental data is obtained via 2 methods: through ControlDesk via the DS1401 box and through the CANbus from DS1401, both sampled at 5Hz. The environment data is given as a vector $v \in \mathbb{R}^{\overline{N}\times 2}$. Each row $v_{i,:}$ corresponds to the left and right environment bounds in meters at time step i with respect to the current heading and position of the car. The bounds correspond to anything that the vehicle should avoid from the edge of the road to obstacles such as vehicles or pedestrians. For example, $v_{i,:} = [6.5, -1.5]$ denotes that at time step i, the car has 6.5 meters on the left before an obstacle or the edge of the road and 1.5 meters on the right before the edge of the road or an obstacle. Therefore, if the road curves to the left without any obstacles, at time step i, $v_{i,:} = [+, +]$, which represents that the environmental bounds at time step i are to the left of the

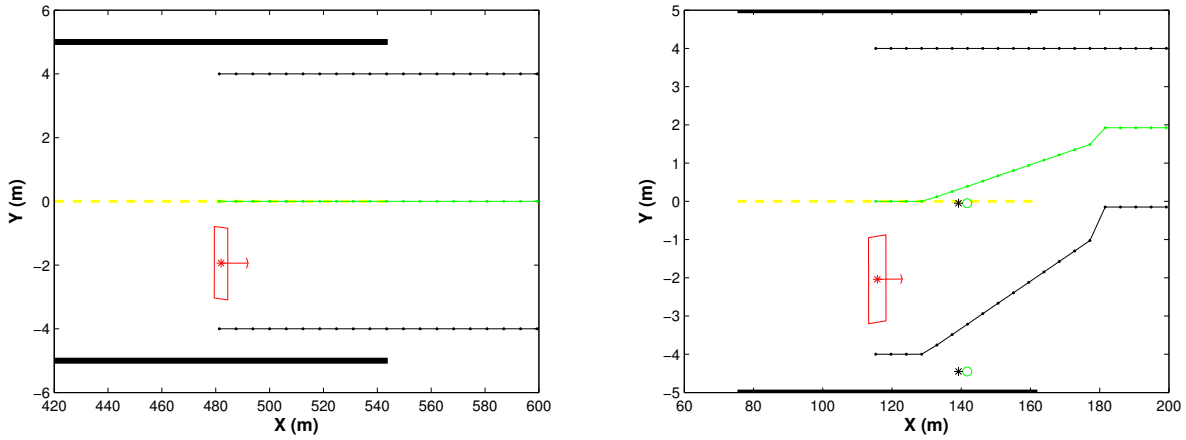

Figure 7: Environmental bounds for a straight road with and without obstacles. The solid black lines represent the edges of the road. The thin black line represents the environment bounds. The green line represents the feature vector. The yellow line represents the center of the two-lane roadway.

current vehicle position.

To allow the driver model to cluster the environment data, we construct a feature vector $\phi_{\overline{N}} \in \mathbb{R}^{\overline{N}}$ by taking the average of each row of v.

$$\phi_{\overline{N}}(i) = \text{average}(v_{i,:})$$

Figure 7 and 8 illustrate examples of the environmental bounds v and the feature vector $\phi_{\overline{N}}$ and Tables 5, 6 represent the numerical values.

$v_{:,1}$	$v_{:,2}$	$\phi_{\overline{N}}(i)$
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5

Table 3: Numerical environmental bounds for a straight road without obstacles

$v_{:,1}$	$v_{:,2}$	$\phi_{\overline{N}}(i)$
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-1.5	2.5
6.5	-0.25	3.125
6.5	1	3.75
6.5	2.25	4.375
6.5	3.5	5
6.5	3.5	5

Table 4: Numerical environmental bounds for a straight road without obstacles with obstacles

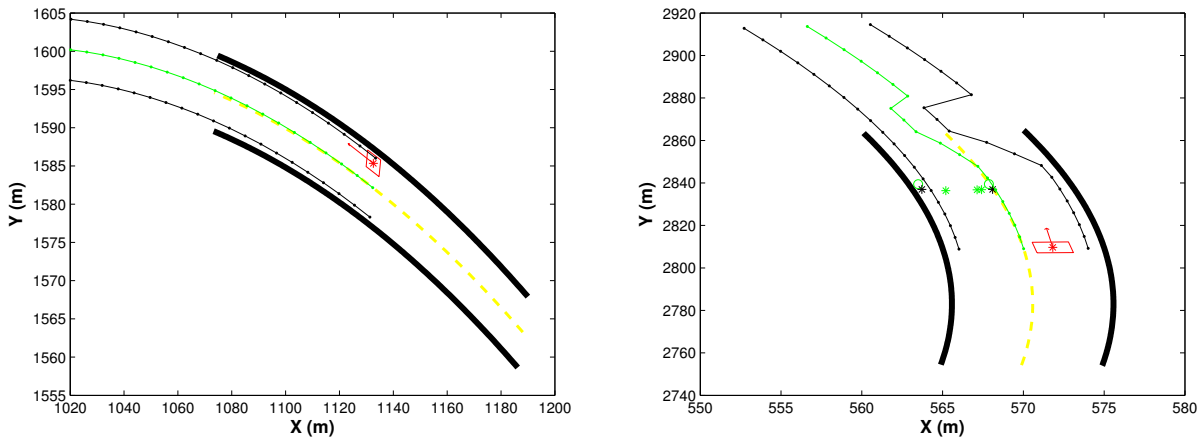

Figure 8: Environmental bounds for a curved road with and without obstacles

$v_{:,1}$	$v_{:,2}$	$\phi_{\overline{N}}(i)$
6.5	-1.5	2.5
6.5	-1.5	2.5
7.5	0.5	4
8.5	1.5	5
9.5	2.5	6
10.5	3.5	7
11.5	4.5	8
12.5	5.5	9

Table 5: Numerical environmental bounds for a curved road without obstacles

$v_{:,1}$	$v_{:,2}$	$\phi_{\overline{N}}(i)$
6.5	-1.5	2.5
6.5	-1.5	2.5
7.5	0.5	4
8.5	2	5.25
9.5	6.5	8
10.5	9.5	10
11.5	4.5	8
12.5	5.5	9

Table 6: Numerical environmental bounds for a curved road without obstacles with obstacles

5.3 Human Posture

The human posture is obtained via the Microsoft Kinect. A Visual studio application was developed using the Microsoft Kinect SDK v1.6 and built primarily off of the SkeletalViewer example code. The 3D positions of both hands, elbows and shoulders after calibration were used to determine the posture at 20Hz. Calibration is performed at the beginning of every experiment to place the coordinate system of the kinect on the driver. The driver gets into driving position, waits for the Kinect to catch onto the driver's skeleton, and holds still for 10 seconds. The kinect uses the neck and shoulder points to create the x-y plane with the cross product representing the z-axis with the neck as the origin. The y-axis would be the line between the shoulders and x-axis would be the cross product of the y and z-axis. An example is shown in Figure 9.

The 3D positions obtained by the kinect are stacked into a feature vector $\theta_{\underline{N}} \in \mathbb{R}^{27\times\overline{N}}$.

5.4 Clustering

After the raw data is processed, we create a driver model using the training data. We define a situation to be a single instance of 2 seconds of driver posture (Section 5.3) and 4 seconds of lookahead of the future environment (Section 5.2). While these situations are technically unlabeled, we can

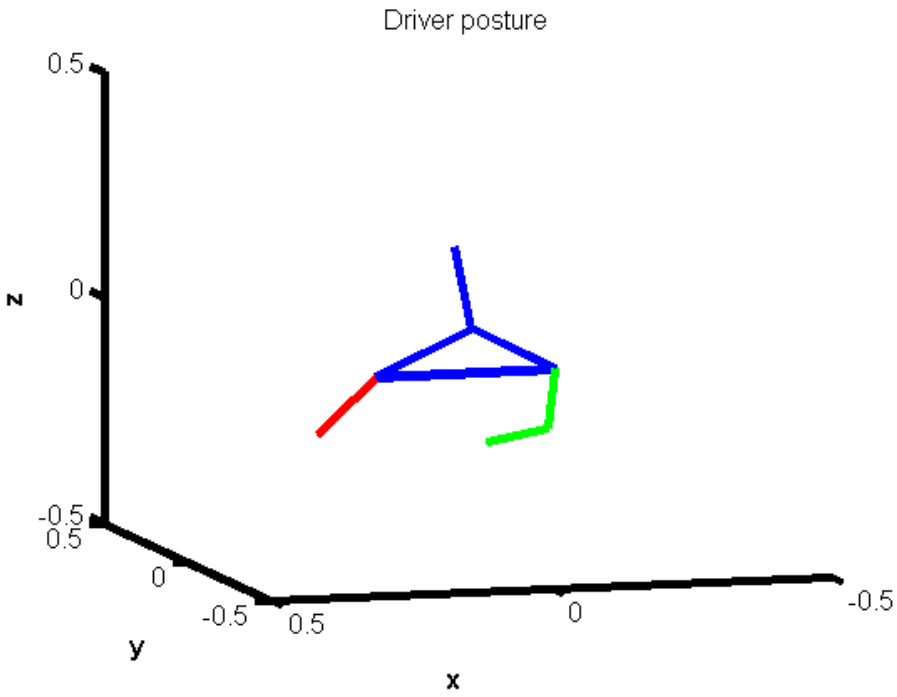

Figure 9: Posture of the driver

view a situation as such: driving straight while attentive, driving straight while distracted, turning while attentive, turning while distracted, etc. The intuition is to cluster similar situations with each other and assigned a set of likely future steering wheel angles and vehicle trajectories for each cluster. Each cluster can also be understood as a mode, defined in Section 3.3. For the remainder of this manuscript, they are equivalent.

This is implemented using k-means clustering in MATLAB. Using the vectors $\phi_{\overline{N}}$, denoting the environment data, and $\theta_{\underline{N}}$, denoting the human posture, we cluster these situations into 100 clusters. After clustering, in each cluster, we obtain the actual 1.2 seconds of future steering wheel angles and vehicle trajectory data and combine them to build the probability distributions described in Section 3.3. Since k-means is the simplest clustering algorithm there is, future work includes developing a better driver model with more advanced machine learning techniques.

An example of 5 of the 100 clusters for a single subject is shown in Figure 11. Notice that in a few of the clusters, the driver's right hand, denoted in blue, is up, indicating that it is most likely off the steering wheel.

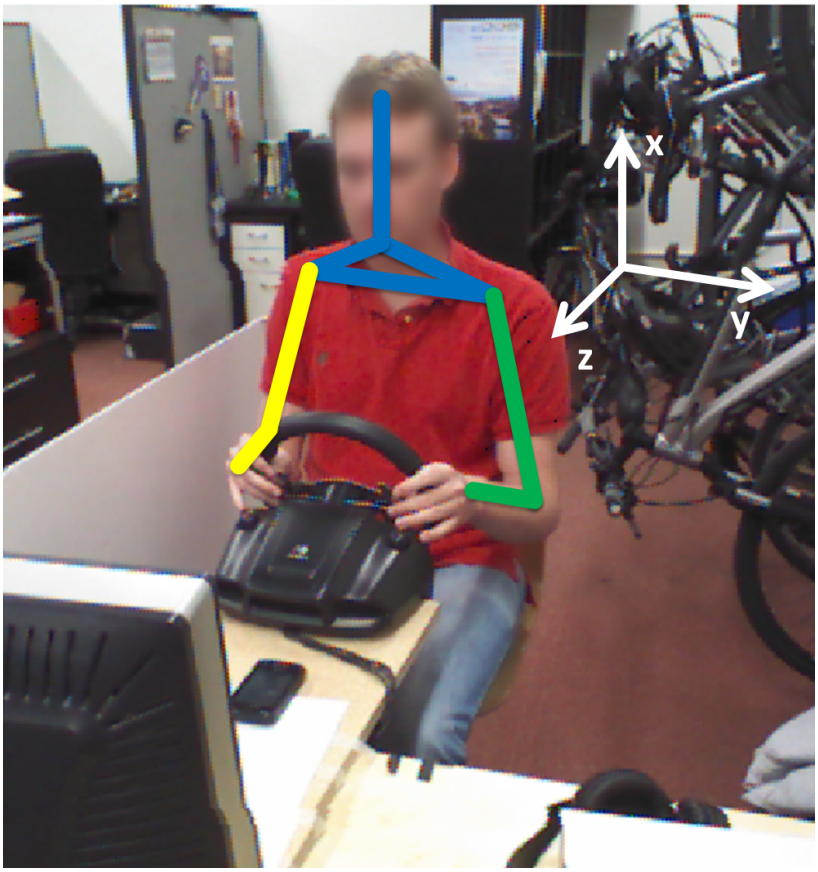

Figure 10: Posture of the driver overlaid on an image with axes

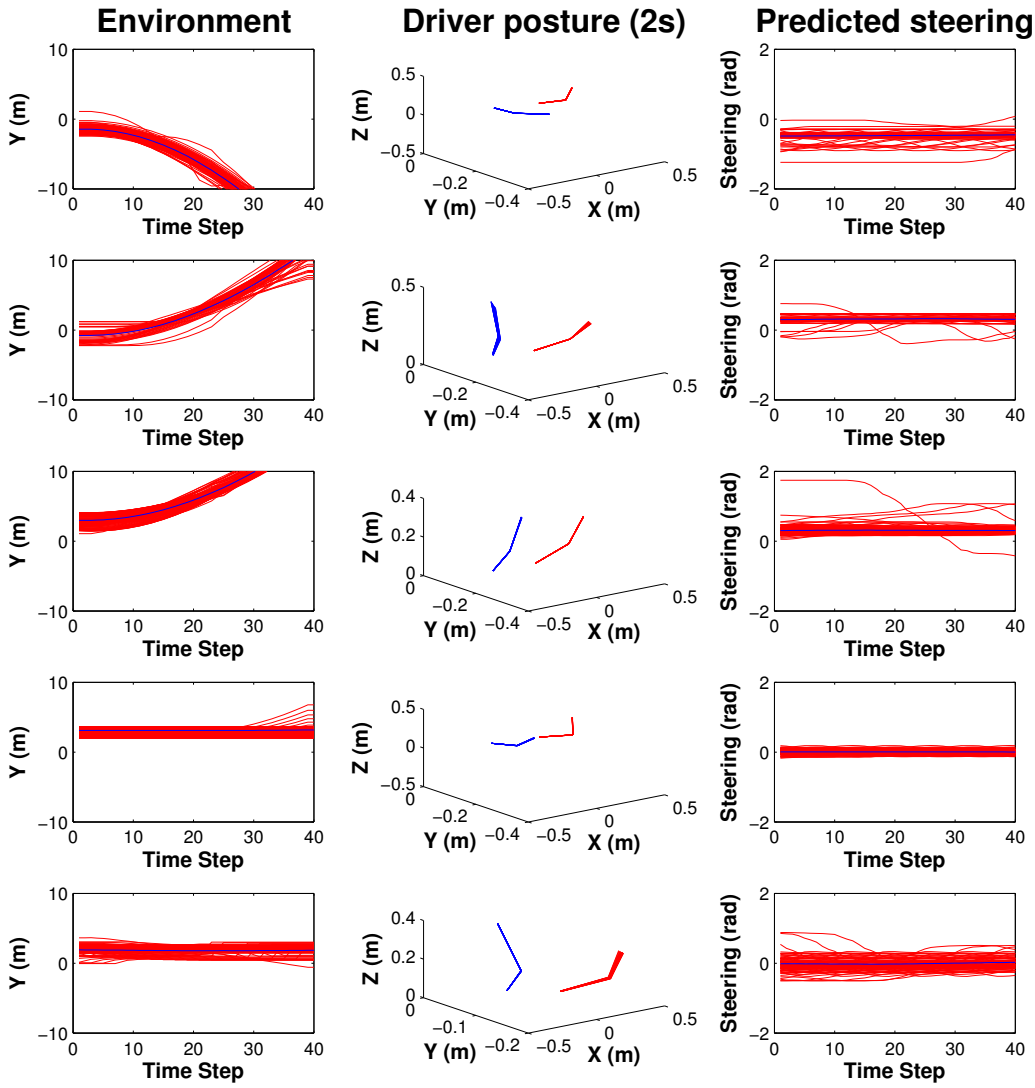

Figure 11: Example of 5 clusters. Each row represents 1 mode (cluster). The first column represents the environmental scenario in the mode. The second column represents the driver posture in the mode for the past 2 seconds. Red denotes the left hand while blue denotes the right hand. The third column represents the predicted steering in that mode, which then can be pushed through the vehicle dynamics to obtain probable vehicle trajectories.

5.5 Real-time driver modeling

After processing the data in MATLAB, the centroid of each cluster and the future bounds and trajectories are written to a text file. A kinect program, based off the SkeletalViewer example, was developed to receive environment bounds from CANbus at 5Hz and steering data from the CANbus at 20Hz, determine which cluster the current situation falls into, and returns the predicted future driver behavior in steering angles over the CANbus to the controller. The entire prediction can be run in under 10 ms.

5.6 Data visualization

In order visualize the inputs and outputs of the system, we coded a visualization script in MATLAB to display the inputs into the driver model as well as the environment data shown in Figure 12.

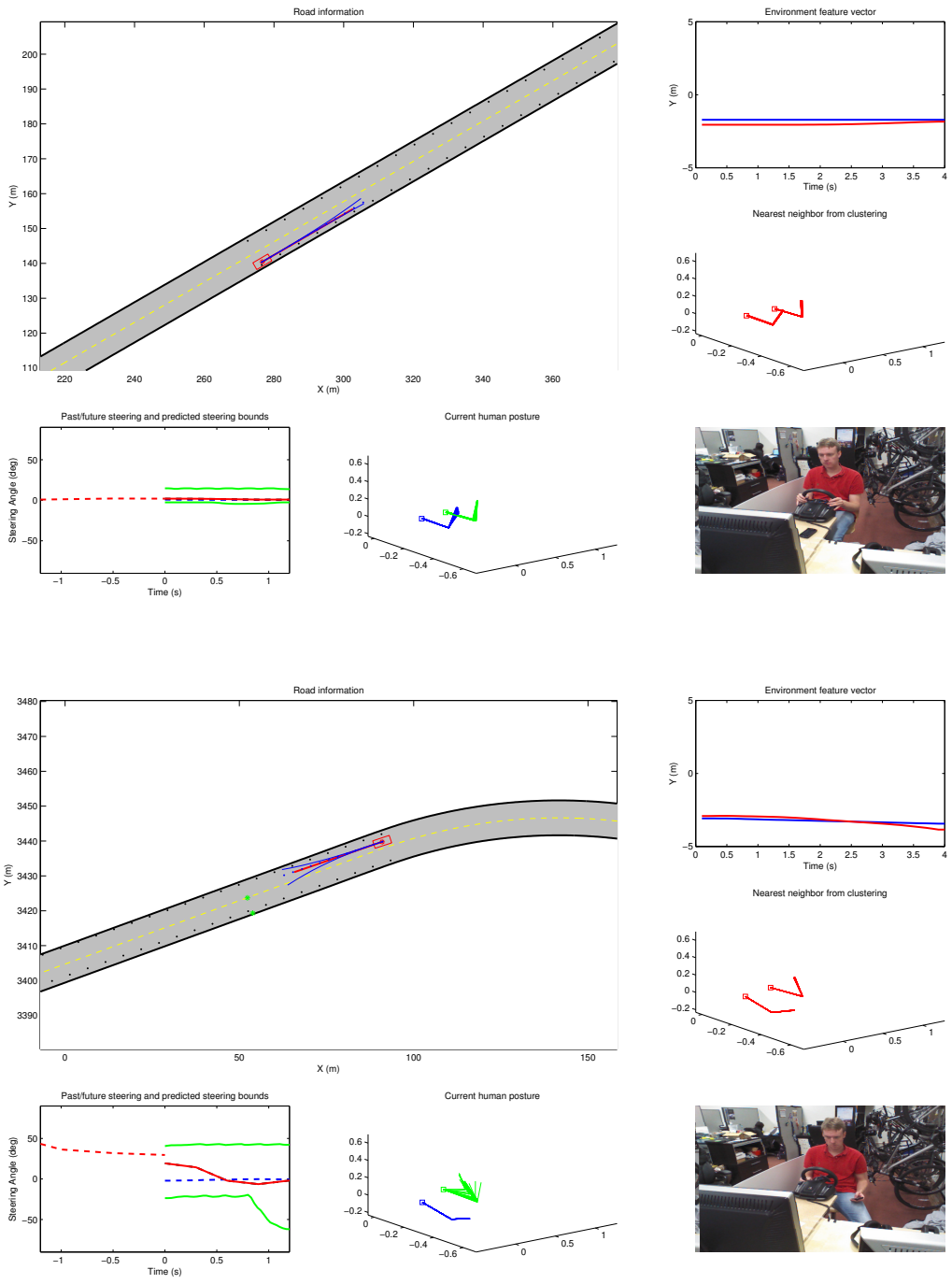

Figure 12: Two instances of the data visualization MATLAB script. Top left displays the road data and prediction set, top right displays the current environment feature vector (blue) and nearest neighbor (red), bottom middle displays the current human posture (left and right arms) and middle right displays the nearest neighbor, bottom left shows the past steering (dotted red), future predicted steering set (green), expected predicted steering (dotted blue) and actual future steering (solid red).

6 Evaluation

In this subsection, we describe how we analyze the experimental data. We ran the experiment on 20 individuals. For each subject, we trained the driver model algorithm using data from the 4 training scenario and tested our performance on the data from the 4 testing scenarios.

6.1 Evaluation of the controller (Scenarios 1-3)

In order to illustrate the utility of the controller in the loop, we ran the driver model in the loop with the MPC controller for the first 3 test scenarios. For the driver model, we considered a vehicle prediction multifunction by clustering the data corresponding to the environment bounds generated for the next $\overline{N}$ time steps beginning at time step i and the driver state $\theta_{i,\underline{N}}$ from the previous $\underline{N}$ time steps ending at time step i. That is, in Equation (14) we set $o_i = (\phi_{i,\overline{N}} : \{0, \ldots, \overline{N}\} \rightarrow \mathbb{R}^{\overline{n}}, \theta_{i,\underline{N}} : \{-\underline{N}, \ldots, 0\} \rightarrow R^{27\times\underline{N}})$. This is meant to illustrate the utility of incorporating the prior observations, the driver state function, and the environment in the semi-autonomous vehicle framework. For this model we fixed the number of modes chosen by the k-means algorithm to 100, set $\overline{N} = 40$, and set $\underline{N} = 40$.

To test the efficacy of the controller, we use the metrics defined in Section 2.2. Once again, the idea is to make sure that the system keeps the vehicle safe but does not assume control too often, otherwise, the vehicle would essentially be a fully autonomous system. We compare this driver model to the reachable set, which metric values we obtain from [37]. From the results shown in Table 7, we see that the prediction set during the experiments with the real-time controller on had an AVPM (accuracy) of 0.8835 and PVPM (precision) of 0.5623 indicating that the predicted set was on average 88% accurate and 56% smaller than the reachable set. Also, when the vehicle was in a highly precarious situation, the controller took over 92% of the time and 96% of the time when the controller took over, it prevented an accident from happening.

	AVPM	PVPM	RVIF	PVIF
Reachable Set	1/0	0	1	0.0031
Driver Model	0.8835/0.0458	0.5623/0.0924	0.9216/0.0739	0.9670/0.0224

Table 7: The mean/std performance of the 3 model semi-autonomous vehicle frameworks with respect to the 4 metrics over all subjects for 50 modes

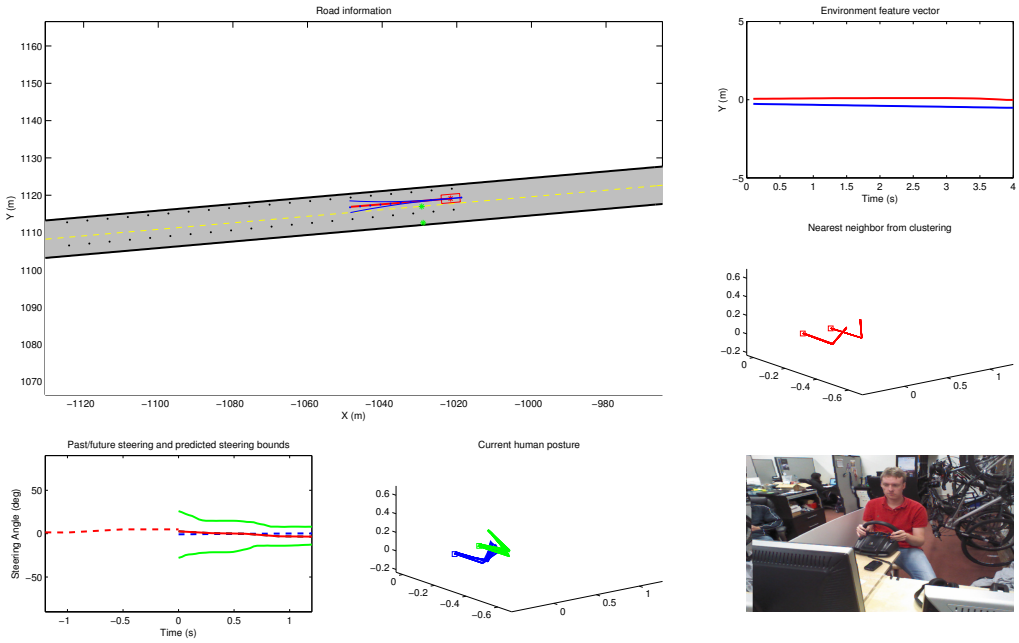

Figure 13: Attentive, Straight driving

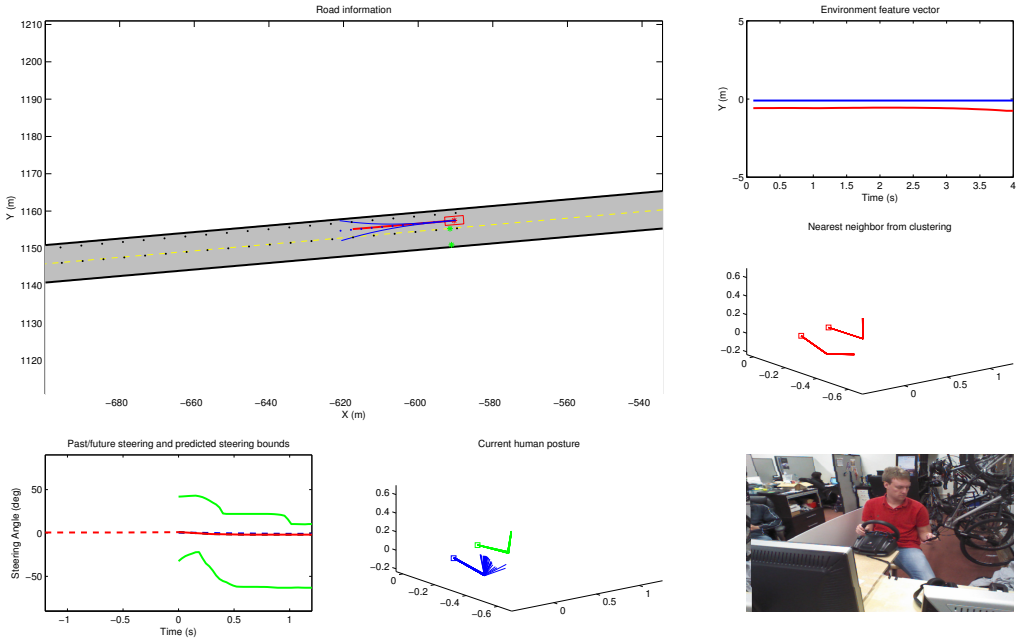

Figure 14: Distracted, Straight driving

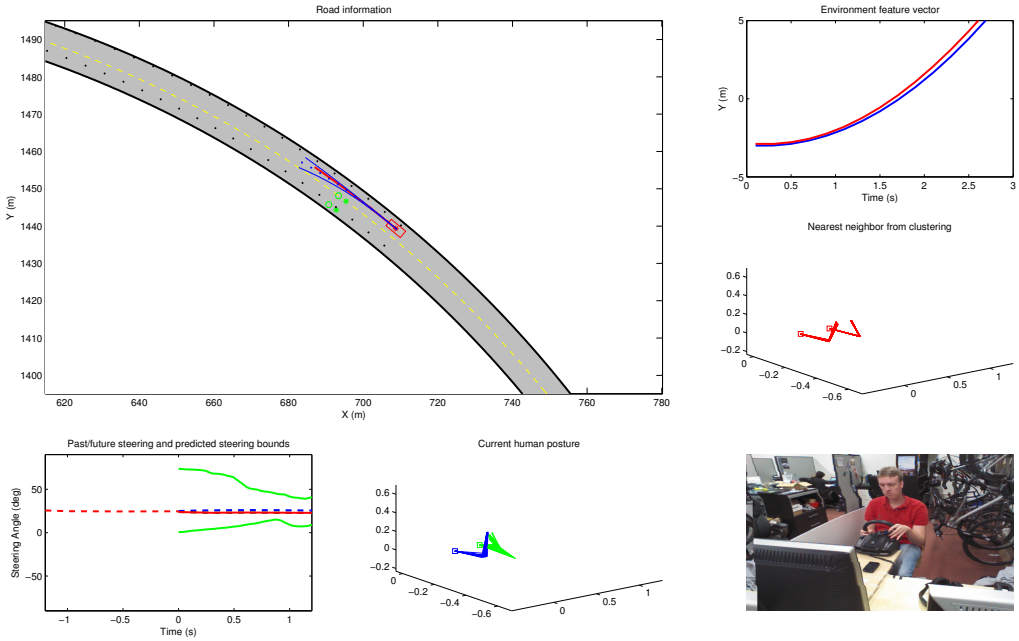

Figure 15: Attentive, Curved driving

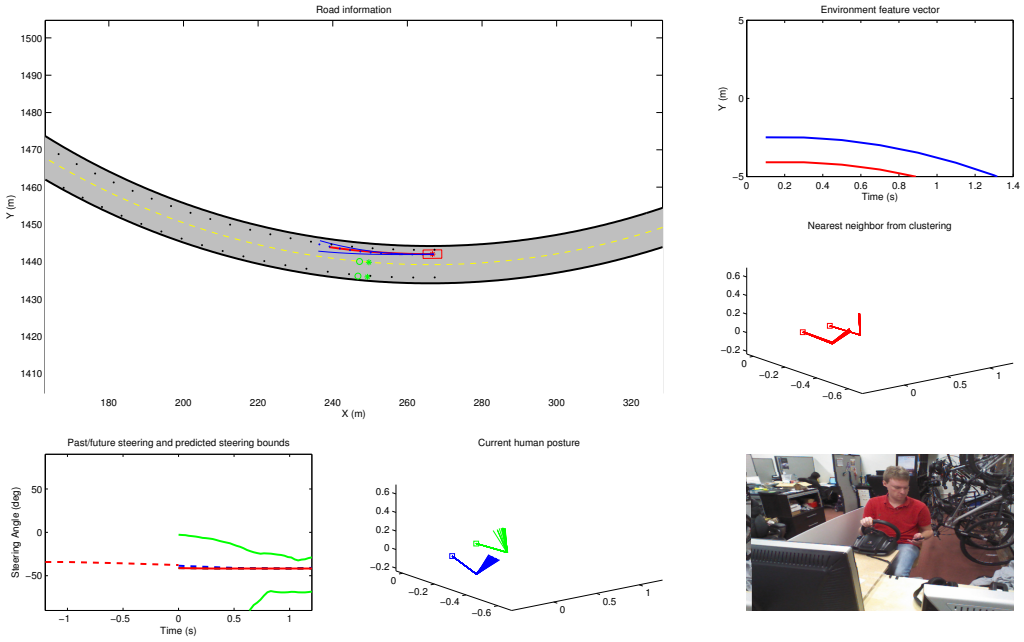

Figure 16: Distracted, Curved driving

6.2 Comparison of models (Scenario 4)

In order to illustrate the utility of our approach, we consider five different implementations of the vehicle prediction multifunction for the last test scenario.

- In *Model 1*, we ignored all prior observations and defined the vehicle prediction multifunction as the reachable set. This serves as an illustration of the baseline performance of existing active safety systems with respect to the AVPM and PVPM. We obtain these metric values from [37].

- In *Model 2*, we constructed the vehicle prediction multifunction by clustering the data corresponding to the environment bounds for the next $\overline{N} + 1$ time steps beginning at time step i. That is, in Equation (14) we set $o_i = (\phi_{i,\overline{N}} : \{0, \ldots, \overline{N}\} \to \mathbb{R}^{\overline{n}})$. This is meant to illustrate the utility of building a vehicle prediction multifunction using the environment bounds.

- In *Model 3*, we constructed the vehicle prediction multifunction by clustering the data corresponding to the vehicle's trajectory for the previous $\underline{N}$ time steps ending at time step i. That

	AVPM	PVPM	RVIF	PVIF
Model 1	$1/0$	0	1	0.0031
Model 2	$0.9468/0.0248$	$0.3861/0.1745$	$0.9217/0.0529$	$0.7185/0.1849$
Model 3	$0.9469/0.0257$	$0.3909/0.1747$	$0.9232/0.0484$	$0.7191/0.1837$
Model 4	$0.9387/0.0302$	$0.4103/0.1588$	$0.9286/0.0516$	$0.7119/0.1865$

Table 8: The mean/std performance of the 6 model semi-autonomous vehicle frameworks with respect to the 4 metrics over all subjects for 50 modes

is, in Equation (14) we set $o_i = (\phi_{i,\underline{N}} : \{-\underline{N}, \ldots, 0\} \to \mathbb{R}^{\underline{n}})$. This is meant to illustrate the utility of building a vehicle prediction multifunction using prior observations without incorporating any observation of driver posture or the environment bounds.

- In *Model 4*, we constructed the vehicle prediction multifunction by clustering the data corresponding to the environment bounds generated for the next $\overline{N} + 1$ time steps beginning at time step i and the driver state function from the previous $\underline{N}+1$ time steps ending at time step i. That is, in Equation (14) we set $o_i = (\phi_{i,\overline{N}} : \{0, \ldots, \overline{N}\} \to \mathbb{R}^{\overline{n}}, \theta_{i,\underline{N}} : \{-\underline{N}, \ldots, 0\} \to R^{27\times\underline{N}})$. This is meant to illustrate the utility of incorporating the prior observations, the driver state function, and the environment in the semi-autonomous vehicle framework.

For models 2-4 we fixed the number of modes chosen by the k-means algorithm to 50 and 100, set $\overline{N} = 40$, and set $\underline{N} = 40$.

The performance of each of the model semi-autonomous vehicle frameworks with respect to our four metrics is illustrated in Table 9. It is immediately clear that by incorporating some information about the driver, we can construct a precise vehicle prediction multifunction without degrading the accuracy considerably.

Consistent across the driver models, adding information regarding the driver reduces the size of the future predicted set by approximately 50%, which may further decrease with a longer time horizon, while maintaining an accuracy of approximately 90%. A RVIF indicates that 90% of instances the driver was unsafe, the control system would have taken over. A PVIF indicates that 70% of the time the system assumed control, the driver would have gotten into an unsafe situation.

	AVPM	PVPM	RVIF	PVIF
Model 1	$1/0$	0	1	0.0031
Model 2	$0.9189/0.0311$	$0.4638/0.1698$	$0.9265/0.0501$	$0.7142/0.1853$
Model 3	$0.9134/0.0318$	$0.4826/0.1650$	$0.91980.0559$	$0.7167/0.1884$
Model 4	$0.8958/0.0342$	$0.4995/0.1492$	$0.9266/0.0454$	$0.7120/0.1879$

Table 9: The mean/std performance of driver model with semi-autonomous vehicle frameworks with respect to the 4 metrics over all subjects for 100 modes

We were able to measure this since the controller did not run in the last scenario.

At this point in time, the results show that providing any type of information is informative for the semi-autonomous controller. However, it is currently inconclusive that providing information on the posture of the driver significantly improves prediction accuracy. However, this does not detract from the narrative that modeling what the driver does in different situations can help the semi-autonomous controller perform better than current systems.

6.3 Issues encountered

These experiments were not without shortcomings. Common subject complaints involved the system not providing enough tactile feedback and that the controller was too “jerky”, a limitation of current hardware. The nonlinear MPC controller ran at 200ms per problem with only 3 inputs over 1.2s, limiting the type of maneuvers and our analysis of the semi-autonomous system. When subjects were driving with the phone in their hand, they often looked up at the road. As the current algorithm used only driver pose to determine driver state, it affected the precision of the driver model. This may be a reason why the driver model 4 which incorporated the human posture was not able to perform better than the driver model 2. We found that the kinect performed poorly with the steering wheel obstructing view of the subject’s hands. In order for the kinect to determine a posture difference, the subjects were asked to text away from the steering wheel which may have altered nominal driving behavior.

7 Conclusion

In this work, we survey the existing literature on distracted driving and human factors, presented an architecture for a provably safe semi-autonomous controller that predicts potential vehicle behavior and intervenes when the vehicle is deemed in trouble, and conducted experiments with human drivers with and without a real-time controller. Using four metrics that we define, we show incorporating a driver model during the construction of the semi-autonomous controller allows for better performance than a semi-autonomous controller that treats the driver as a disturbance. This semi-autonomous framework is modular and easily allows the use of different driver models and controllers.

In order to realize this framework in practical situations, future work involves improving the driver model, employing stochastic MPC to take advantage of the gaussian distributions from the driver model, implementing a system in a vehicle that can perform real-time robust articulated human tracking, and determining how to safely give control back controller to the driver. Systems such as those presented in [15] may serve as systems to implement in our framework to improve on the driver model and semi-autonomous vehicular safety systems. Immediate work includes transitioning the simulator setup to a real car, using OpenCV and stereo cameras to determine driver posture, eye-trackers to determine where the driver is looking, and capacitive touch sensors on the steering wheel to determine a driver model.

While car companies continue to look into the semi-autonomous safety systems, these systems mostly consider only the outside environment. Our work proposes that we should also look inside the vehicle, at the driver, to predict future driving behavior and shows that it can improve vehicular safety systems.

References

[1] CarSim, Mechanical Simulation.

[2] dSPACE Inc.

[3] Mobileye.

[4] NHTSA Data Resource Website.

[5] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *International Journal of Vehicle Autonomous Systems*, 8(2):190–216, 2010.

[6] D. Ascone, T. Lindsey, and C. Varghese. An Examination of Driver Distraction as Recorded in NHTSA Databases. *Online, September*, 2009.

[7] D. Basacik, N. Reed, and R. Robbins. Smartphone use while driving: a simulator study. *Smartphone use while driving: a simulator study*, 1(1):1–67, Mar. 2012.

[8] D. Basacik and A. Stevens. Scoping study of driver distraction. *Transport Research Laboratory. Road Safety Research Report*, 95, 2008.

[9] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming – the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[10] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez. Real-time system for monitoring driver vigilance. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):63–77, 2006.

[11] E. Coelingh, L. Jakobsson, H. Lind, and M. Lindman. Collision Warning With Auto Brake-A Real-Life Safety Perspective, 2007.

[12] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Pr, 2000.

[13] E. D. Dickmanns, R. Behringe, D. Dickmanns, T. Hildebrand, M. Maure, F. Thomanek, and J. Schiehlen. The seeing passenger car 'VaMoRs-P'. In *Intelligent Vehicles Symposium*, 1994.

[14] E. D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. *10th Triennial World Congress of the International Federation of Automatic Control*, 1987.

[15] A. Doshi and M. Trivedi. On the Roles of Eye Gaze and Head Dynamics in Predicting Driver's Intent to Change Lanes. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):453–462, Sept. 2009.

[16] A. Doshi and M. M. Trivedi. Tactical driver behavior prediction and intent inference: A review. *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1892–1897, Oct. 2011.

[17] J. Frankel, L. Alvarez, R. Horowitz, and P. Li. Robust Platoon Maneuvers for AVHS, 1994.

[18] A. Gray, M. Ali, Y. Gao, J. K. Hedrick, and F. Borrelli. Semi-Autonomous Vehicle Control for Road Departure and Obstacle Avoidance. *IFAC Control of Transportation Systems*, 2012.

[19] E. Guizzo. Toyota's Semi-Autonomous Car Will Keep You Safe, 2013.

[20] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[21] R. Y. Hindiyeh and J. C. Gerdes. Equilibrium Analysis of Drifting Vehicles for Control Design. *Dynamic Systems and Control Conference*, 2009.

[22] M. Jabon, J. Bailenson, E. Pontikakis, L. Takayama, and C. Nass. Facial Expression Analysis for Predicting Unsafe Driving Behavior. *IEEE Transactions on Pervasive Computing*, 2010.

[23] S. G. Klauer, U. S. N. H. T. S. Administration, V. P. Institute, and S. U. T. Institute. *The Impact of Driver Inattention on Near-crash/crash Risk: An Analysis Using the 100-Car Naturalistic Driving Study. Data*. National Highway Traffic Safety Administration, 2006.

[24] H. Kress-Gazit, D. C. Conner, H. Choset, A. A. Rizzi, and G. J. Pappas. Courteous cars. *Robotics & Automation Magazine, IEEE*, 15(1):30–38, 2008.

[25] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu. A Driver Behavior Recognition Method Based on a Driver Model Framework. *Society of Automotive Engineers*, 2000.

[26] N. Oliver and A. P. Pentland. Graphical models for driver behavior recognition in a smartcar. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 7–12. IEEE, 2000.

[27] D. Redelmeier and R. Tibshirani. Association between Cellular-Telephone Calls and Motor Vehicle Collisions. *New England Journal of Medicine*, 336(7), 1997.

[28] M. A. Regan, J. D. Lee, and K. L. Young. *Driver distraction: Theory, effects, and mitigation*. CRC, 2008.

[29] A. Sathyanarayana, S. Nageswaren, H. Ghasemzadeh, R. Jafari, and J. H. L. Hansen. Body sensor networks for driver distraction identification. In *IEEE International Conference on Vehicular Electronics and Safety*, pages 120–125. IEEE, 2008.

[30] J. C. Stutts and W. W. Hunter. Driver inattention, driver distraction and traffic crashes. *ITE Journal*, 73(7):34–45, 2003.

[31] R. Subbarao and P. Meer. Nonlinear mean shift over riemannian manifolds. *International journal of computer vision*, 84(1):1–20, 2009.

[32] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer. Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, May 1988.

[33] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, and Others. Stanley: The robot that won the DARPA Grand Challenge. *The 2005 DARPA Grand Challenge*, pages 1–43, 2007.

[34] M. M. Trivedi, T. Gandhi, and J. McCall. Looking-In and Looking-Out of a Vehicle: Computer-Vision-Based Enhanced Vehicle Safety. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):108–120, Mar. 2007.

[35] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, and Others. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[36] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207, 1993.

[37] R. Vasudevan, V. Shia, Y. Gao, R. Cervera-Navarro, R. Bajcsy, and F. Borrelli. Safe semi-autonomous control with enhanced driver modeling. In *American Control Conference*, 2012.

[38] E. Wahlstrom, O. Masoud, and N. Papanikolopoulos. Vision-based methods for driver monitoring. In *IEEE Proceedings on Intelligent Transportation Systems*, volume 2, pages 903–908. IEEE, 2003.

[39] A. M. Waxman, J. Lemoigne, L. S. Davis, B. Srinivasan, and T. R. Kushner. A Visual Navigation System for Autonomous Land Vehicles. *Journal of Robotics and Automation*, (2):124–141, 1987.