MONITORING AND CONTROL OF EXTREME CONGESTION EVENTS IN ROAD TRAFFIC NETWORKS

BY

SUDEEP GOWRISHANKAR

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Advisers:

Assistant Professor Daniel B. Work
Professor Geir E. Dullerud
Abstract

This thesis addresses the problem of traffic management during extreme traffic congestion events. In particular, it describes the development of a new traffic sensing technology called TrafficTurk that is capable of being deployed through multiple temporary sensors quickly over an entire road network. The development of the technology is motivated by the enormous cost to society that is associated with extreme traffic congestion caused by pre-planned events such as sporting and cultural events, extreme weather and natural disasters. The eventual goal of the technology is to provide a comprehensive system for traffic management of extreme congestion events through real-time data collection and processing. Therefore, detailed explanations of data processing algorithms that are required to enable further development of the system are also provided. Specifically, a method to estimate traffic controller strategies at intersections in a road network using inverse optimal control and a method to normalize traffic data collected using TrafficTurk are explained in detail. Additionally, practical implementation knowledge gained thus far through extensive field testing of the system is also discussed.
A number of people have played very important roles in making this work possible. First, I would like to thank my adviser, Prof. Daniel Work, for always having the time and patience to discuss and guide me through all the different challenges of earning a master’s degree. In addition, I would like to thank the entire TrafficTurk team who made the sometimes grueling process of developing and deploying a technology a thoroughly enjoyable experience. Brian Donovan, Mostafa Reisi Gahrooei, Meng Han, Jonathan Que, Bhinav Sura and Camilo Vega, it would not be an understatement to say that I have learnt and grown immensely through our shared experiences and it has been an honor to work with all of you. I would also like to thank Prof. Geir Dullerud, who facilitated my research, and Prof. Viswanathan, who provided me with diverse learning experiences throughout my graduate degree. And finally, a big thank you to my parents, family and friends for all the sacrifices you have made for me and all the support you have provided over the years.
# Table of Contents

1 Introduction ................................................................. 1  
1.1 Motivation ............................................................... 1

2 TrafficTurk - A smart-phone based traffic sensing technology  4  
2.1 Motivation behind TrafficTurk as a smart-phone application  . 4  
2.2 TrafficTurk client-side application design ......................... 6  
2.3 TrafficTurk back-end system design ............................... 9  
2.4 Algorithms and data processing ................................. 10

3 Estimating Traffic Control Strategies with Inverse Optimal Control ........................................... 12  
3.1 Introduction ............................................................... 12  
3.2 Intersection traffic model and optimal signal control .......... 15  
3.3 Recovering switching control objectives via inverse optimal control 18  
3.4 Discussion of feasibility of solutions and approximate optimality 23  
3.5 Experiments ............................................................... 25  
3.6 Summary ................................................................. 41

4 Large-scale field deployments of TrafficTurk .................. 42  
4.1 Overview ................................................................. 42  
4.2 Data Collection at the Farm Progress Show ...................... 43  
4.3 Potential improvements for future deployments ................. 55  
4.4 Traffic Data Analysis .................................................... 56  
4.5 Functionality Analysis .................................................. 72  
4.6 Scalability ............................................................... 77  
4.7 Potential Future Uses of TrafficTurk ............................ 79
5 The Future of Traffic Monitoring and Control of Extreme Congestion Events ........................................ 81

Bibliography ....................................................... 83
1

Introduction

1.1 Motivation

Traffic networks are, have been, and will continue to be for the conceivable future, the lifelines of modern society. Economies rely on people, goods and services traveling from one place to another to function and prosper. However, with the increased demand on our traffic networks due to more advanced, affordable technology and the effects of globalization, congestion of traffic networks has become a serious problem facing the world today. Road traffic networks in particular are more congested than ever before and it is estimated that in the United States alone, the cost of road traffic congestion was $121 billion in 2011. This included a total of 2.9 billion gallons of fuel and 5.5 billion hours of travel delay [1]. This work focuses on the development of solutions to the worst examples of congestion, referred to as extreme congestion events, that cause enormous disruption in terms of delay and safety. These extreme congestion events occur mainly due to sporting or cultural events, extreme weather and natural disasters.

A recent report by the Federal Highway Administration identifies seven main causes of road traffic congestion - traffic incidents, work zones, weather, fluctuations in normal traffic, special events, traffic control devices and physical bottlenecks [2]. Out of these, pre-planned special events in the United States alone are estimated to cause between 93-187 million hours of delay, with associated costs ranging between $1.7 and $3.4 billion [3]. Even though there are over 24,000 special events that draw over 600 million people annually,
not much research has been done so far to understand and manage the traffic induced by these events. In addition, extreme weather and natural disasters are generally inevitable circumstances, and being able to understand and manage traffic reliably during these conditions is very important to improve safety and potentially reduce the loss of human life.

One factor in particular makes studying and managing extreme congestion caused by special events, extreme weather and natural disasters difficult - the lack of traffic sensing infrastructure. Apart from a few exceptions, dedicated traffic sensing infrastructure is very limited and as a result, collecting traffic data over a large urban traffic network is generally not possible nor feasible. Some efforts to gather traffic data despite the lack of sensing infrastructure have included GPS data from smart-phones and navigation devices. However, even this data source is temporally-spatially sparse and as a result, there has been a reliance on statistical models to predict traffic conditions.

The challenge with using statistical models to predict traffic conditions during extreme congestion events is that by their nature, extreme congestion events often change the model of traffic flow. For example, events such as marathons result in road closures and changed behavior of traffic control devices. Often, policemen are sent to manually over-ride the traffic signal controller or hand-direct traffic during special events. These events also result in changed travel demands since significantly more people are going to the start line of a marathon, for example. There are similar effects when extreme weather or natural disasters occur. For example, Hurricane Sandy that hit the east-coast of the United States in 2012 flooded many roads and rendered them unusable and damaged traffic signals. There was also a shortage of vehicle fuel that created atypical demand for roads with gas stations located on them.

Therefore, an immediate need for traffic management during extreme congestion events is the ability to monitor traffic in a reliable, easily deployable and yet cost effective manner. Dedicated sensing infrastructure such as inductive loop detectors and cameras are very expensive to install, set up and maintain over the whole traffic network and therefore, they are not feasible solutions to overcome the insufficient amount of available traffic data.

Thus, this thesis explains the development of a new system to monitor and manage traffic during extreme congestion events and is organized as follows.
Chapter 2 introduces a new technology called TrafficTurk that can be used to collect traffic data in situations where a fast and inexpensive deployment of traffic sensors is required. Chapter 3 describes a method to estimate the traffic control strategies at intersections in the road network based on data collected by TrafficTurk. Then, in Chapter 4, a detailed report of a large-scale deployment of TrafficTurk is presented. This chapter is a useful description of the various technical and logistical issues faced and solved during this deployment. It also contains a general methodology for deploying TrafficTurk in the field and the practical solutions that were developed and implemented since the birth of the technology. Finally, a discussion about the future of traffic management during extreme congestion events is provided to inform future work in the area.

Specifically, the main contributions of this thesis are the development of an inverse optimal control method to estimate the traffic control strategies of traffic controllers using TrafficTurk data and the experimental design, implementation and analysis of the technology in the field.
2

TrafficTurk - A smart-phone based traffic sensing technology

The motivation behind the development of TrafficTurk is to create a traffic monitoring system that is capable of being deployed anywhere in the world within a short time-frame and for a short period of time. This requires the use of technology and resources that are widely available and accessible. Therefore, TrafficTurk uses a smart-phone application that lets humans collect traffic data. The motivations behind this method of data collection are explained in the following section.

2.1 Motivation behind TrafficTurk as a smart-phone application

The name TrafficTurk is inspired by an 18th century chess playing machine named the Mechanical Turk [4]. This machine was famous for beating some of the world’s best chess players much before the birth of computers and artificial intelligence. However, it was later revealed that a person hid inside the machine and pulled levers to play the game and compete against human chess players. (The machine has also inspired the name of Amazon.com’s human-based web service.) In a similar way, TrafficTurk utilizes humans to collect traffic data.

There are a couple of reasons to use humans to collect traffic data. Firstly,
conventional traffic sensors such as inductive loops and cameras have high installation costs and since quite often, extreme congestion events cause congestion in parts of the network that are normally uncongested, it is not economically feasible nor viable to cover an entire network with conventional sensors. With human sensors, the installation costs are minimal since more and more people are owning smart-phones. The zero cost of the devices (people already own smart-phones and they use their own phone while collecting traffic data), downloading the application and a relatively small cost of hiring people to collect traffic data make TrafficTurk an inexpensive sensor. Field tests have shown that it is generally sufficient to pay an individual between $15-20 for every hour of data collection.

The second reason to use humans to collect traffic data is their ability to easily distinguish between modes of transportation. In a future version of the TrafficTurk application, users will be able to accurately identify not only vehicles and pedestrians, but also the mode of transport being used. This feature is currently under development, but the rationale behind it is to be able to collect traffic data in environments where traffic is heterogeneous (cars, trucks, bicycles, motorbikes, etc.).

Finally, transportation engineers have collected traffic data in locations without dedicated sensing infrastructure with humans for a long time and it is an established sensing method. Traditionally, they have used devices called turning movement counters. These devices generally have a number of buttons that represent vehicular and pedestrian movements at an intersection (Figure 2.1). An individual equipped with a turning movement counter goes to an intersection and presses the appropriate buttons as they see vehicles and pedestrians passing the intersection. These data are called turning movement counts and they are used to study, understand, and plan for traffic at an intersection.

TrafficTurk also uses humans to collect turning movement counts but aims to overcome some of the deficiencies of the traditional turning movement counter. Firstly, traditional turning movement counters are not generally capable of streaming real-time data to a central location for processing; the data transfer from the device to a database is often done offline. This does not allow for any kind of real-time traffic prediction and control, which can be
very useful during extreme congestion events to route emergency vehicles and to create adaptive control strategies.

Secondly, since turning movement counters are specialized for collecting turning movement counts and since they are relatively expensive (over $300), they are not ubiquitous. This makes using the traditional turning movement counter unsuitable for fast deployment in different locations around the world because the devices may not be available in the required quantity to monitor an entire traffic network. TrafficTurk was developed as a smart-phone application due to the ever growing reach and ubiquity of smart-phones and their connectivity to the internet. In addition, we found that the touchscreen interface of most smart-phones allows for an intuitive swipe based data entry method that is designed to minimize human error in the collected data. Figure 2.1 shows the data collection interface used in the TrafficTurk application.

2.2 TrafficTurk client-side application design

Parts of this section are excerpts from a white paper that is in preparation for submission to the Transportation Research Board Conference 2014 [6].

**Client-side application design** While the smart-phone platform provides many hardware and communications benefits, it presents several new design challenges. Our first attempt at designing a smart-phone based turning move-
ment counter directly emulated the traditional turning movement counters, complete with a 12 button interface. However it was soon found that the interface was quite difficult to use due to the small size of smart-phone screens, and the lack of a tactile response. Unlike dedicated boards, it was difficult to verify that the correct button was pressed, without looking at the phone screen.

Given that most smart-phones are manufactured with high precision touch-screens, our second design focused on using swipes (gestures) on the screen to record turning movement counts. For example, by tracing a finger along a path indicated by the arrows in Figure 2.1, the user is able to record each of the 12 movements, along with the timestamp of the count. Accurate swipe recognition is also a non-trivial problem, due to the wide variety of screen sizes manufactured on the Android platform. Moreover, the wide variety of hand sizes, combined with varying screen and resolution sizes, meant a reliable swipe recognizer needed to be developed. The result was a simple interface that allows fast and accurate data collection without the need to look at the screen.

A second design challenge was created by the desire to use the TrafficTurk for real-time monitoring applications. First, this means the application must communicate with a back-end server regularly to transmit the latest counting information. Second, it means the application and the back-end server must agree on exactly which intersection the application is recording counts. The simple solution to this problem was to have the server keep track of the network topology, and share with the application the locations where the phone may record counts. Once the phone receives a list of intersections at which counts can be collected (see Figure 2.2), the user selects the intersection, and the compass is used to determine the alignment of the user, relative to the intersection.

Synchronization of this map and count information proved harder than one might expect, precisely because extreme congestion events also cause extreme congestion on the cell phone network infrastructure. For example, in the case of a natural disaster, the network may be completely unavailable. Designing an application to support synchronization of all information about intersections to display on the phone proved quite difficult, because of the wide range of
communication network quality. Ultimately, we insisted that the application appear to be completely functional to users, regardless of the communication capabilities with the server. Moreover, the application is required to intelligently cache information about the network topology, and opportunistically transmit data to the server whenever a connection with the server can be established.

Because it is difficult to obtain information about traffic signal phases [7], and because signal phase timings are relevant for predicting how the system will respond to future traffic demands, we decided that the application should also collect information on the traffic signal phases. In a preliminary version of the application, we created a separate data entry mode within the application, where users could directly record the traffic signal phase. This was problematic because it was impossible for users to simultaneously record vehicle counts and phase changes reliably. Instead, we made a design decision to estimate the signal phases directly from the turning movement counts. In this way, we keep the application design simple, and we do not require the user to input any data that the application itself can deduce from the recorded data.

Another important aspect of the application design is the user interface. The application has been carefully designed to be as intuitive as possible. We attempt to present the smallest amount of written information, buttons, and interactions to the user at every step and provide feedback whenever necessary. The chosen aesthetics and graphic styles are designed to present information clearly and a game-like experience is provided to keep users engaged.
2.3 TrafficTurk back-end system design

TrafficTurk relies on a comprehensive software stack in the back-end to be able to do everything from log users into the app, send them map information, receive collected data, etc. This part of the system will not be explained in detail in this thesis. Instead, a short summary of the main features of the back-end system and its functionalities will be described, in order to provide a background for the comprehension of the main contributions of this thesis.

The main hardware component of the back-end system is a server. It houses a database for long-term storage of data and runs a communication protocol that manages connections from multiple clients (smart-phones). The database is an open-source object relational database and the communication protocol is a custom protocol that was developed by Brian Donovan, a member of the TrafficTurk team. The main function of the communication protocol is to transfer data between the database and the client in both directions. The database stores information about the physical road network in the form of a processed world map. Only the required sections of the world map are loaded into the database due to storage space and querying speed considerations.

The world map data is sourced from a crowd-sourced mapping engine called OpenStreetMap [8]. The raw map data is pre-processed and stored in the database in the form of intersections (nodes) and roads (links). The pre-processing is required in order to only store the data that TrafficTurk requires in the database and in order to ensure that intersections and roads are correctly created. These nodes and links are sent to the client when requested in order to geo-code the traffic data being collected. The geo-coding helps to exactly identity each vehicle’s movement at an intersection.

Once traffic data is collected, the client sends it over the internet to the database through the communication protocol. The protocol uses TCP sockets and a system of acknowledgments to ensure that each and every data-point that is collected by a client is saved in the database regardless of whether an internet connection is immediately available or not. The application and the server work together to opportunistically send and receive the collected data. This data is then stored for long-term storage in the database.
2.4 Algorithms and data processing

Another crucial component of the TrafficTurk system is the data processing engine that uses methods and algorithms to analyze the data. These methods are being continuously developed to support the system. The eventual goal of TrafficTurk is to become a complete traffic monitoring system for extreme congestion events that includes data collection, traffic prediction and possibly even traffic control. However, a few pre-processing algorithms and methods were needed to solve a number of critical problems with the collected data before any traffic prediction or control can be done. This section describes an algorithm to estimate the traffic signal phases from TrafficTurk data as a background for the detailed description of estimating traffic control strategies with inverse optimal control in the next chapter.

2.4.1 Signal phase estimation

A goal of the TrafficTurk system is to be able to estimate traffic flows based on models and data. In order to do this, we not only need turning movement counts, but also knowledge of the traffic signal controllers and their phases. This is because the vehicular flow on a road is determined by both the number of cars on the road and the speed that they are traveling. Therefore, vehicular flow can be zero when there are no cars on the road and when there is a traffic jam and vehicles are not moving. In order to distinguish between these situations, knowledge of traffic signal phasing is important.

However, obtaining this information from traffic agencies can be practically difficult at times. Furthermore, collecting this information through the application by asking the users to enter the phasing information in addition to the turning movement counts is not feasible from a usability perspective. Early prototypes of the TrafficTurk application implemented this exactly and we found that it hindered their ability to collect turning movement counts accurately. Therefore, a signal phase estimation algorithm was developed to extract the phasing information directly from the turning movement counts.

The details of the algorithm are described in [9]. The algorithm models traffic flow through an intersection as a Hidden Markov Model and the turning movement counts are treated as observations from different hidden states,
which in this case, are just the phases of the traffic signal. These needed to be inferred for each turning movement count. The algorithm uses a learning and inference algorithm to estimate the phases at an intersection with high accuracy.
3

Estimating Traffic Control Strategies with Inverse Optimal Control

This chapter is an excerpt of an article that is in preparation for submission to the IEEE Transactions on Intelligent Transportation Systems, December 2013. A conference version of the same article was submitted and accepted to the IEEE Intelligent Transportation Systems Conference, 2013 [10].

3.1 Introduction

3.1.1 Motivation

Urban traffic estimation and forecasting is a challenging problem, especially when knowledge of the traffic signal control policy is unknown to the estimator. Often, signal control strategies are known by the local authorities, but the information is difficult to obtain at larger scales. In other cases, the control strategies are based on proprietary algorithms, and thus cannot be obtained explicitly, even from the managing agencies.

During extreme congestion events such as sporting events and natural disasters, the traffic signal control is performed manually by traffic management police officers. Some reasons for manual control of traffic include the failure of traffic lights during disasters due to power outages, and the fact that the con-
troller itself may not be optimized for extreme congestion events due to their rare occurrence. In each of the above situations, it is important to be able to quickly learn the control strategy of the controller (whether human or not) so that traffic prediction and estimation systems can integrate information on how the flow is regulated.

Therefore, in this chapter, we propose a method to address the following problem. *Is it possible to learn the control objective of a traffic signal, by observing the queue lengths at the intersection and the corresponding control actions?*

### 3.1.2 Related work

The problem of estimating properties of the traffic signal from sensor data has been examined by several authors. For example, [11] developed a method to estimate queue lengths at signalized intersections using travel times through the intersection, collected from mobile GPS sensors. Hofleitner et al. presented an unsupervised classification algorithm to infer the existence of a traffic signal on a road segment using sparse probe vehicle data [12]. Another related project is *SignalGuru* [7], which is a *Green Light Optimal Speed Advisory* (GLOSA) system that uses the camera and communication capabilities of windshield mounted smart-phones to advise drivers about the optimal driving speeds in order avoid stopping at intersections. The system builds a database of fixed time signals, while adaptive signals are predicted with a support vector machine using a week long log of the adaptive signals. Our work differs from *SignalGuru*, both in the sensing (TrafficTurk measures vehicle maneuvers, from which the signal phase timing must be inferred [9], and not directly the traffic signal), and in the control objective estimation approach. The *SMART-SIGNAL* [13] system is another initiative that aims to collect high-resolution data from signalized intersections and use it to infer useful knowledge of the traffic system. This system communicates valuable traffic information which is often only available at the roadside signal controller, and may significantly improve the information available to traffic estimation systems in the future.

This work formalizes the problem of estimating the control strategies as an *inverse optimal control* (IOC) problem. Unlike optimal control, which
computes a control policy that maximizes some performance objective under constraints [14], the inverse optimal control problem aims to recover the unknown objective function given realizations of the system trajectory which are assumed to be optimal. This is a useful concept in our problem since we can use it to find an objective function under which the true system’s policy is optimal and hence recovering the control objective of the true system. The inverse optimal control problem was studied as early as [15], and recently in the machine learning community as a related problem of inverse reinforcement learning (IRL). For example, Ng and Russell [16], Abbeel and Ng [17] have proposed methods that span applications including learning the control strategy for helicopter acrobatics and bipedal robots. In the realm of transportation systems, inverse reinforcement learning has been studied in relation to autonomous parking lot navigation [18], navigation and driving behaviors [19], helicopter flight [20], and hybrid vehicle fuel efficiency [21] among others.

However, as [22] points out, many approaches to inverse optimal control include the repeated solving of an optimal control problem within the inverse reinforcement learning framework and tend to be computationally intensive. Keshavarz et al. [23] propose a method to significantly reduce the computational requirements by posing the inverse optimal control problem as a convex optimization problem. Aghasadeghi et al. [22] extend the idea in [23] to solve an inverse optimal control problem for a hybrid system with impacts, which also inspired the approach taken in this paper.

In particular, optimal control approaches to traffic have been widely studied and part of our work is derived from the work of De Schutter [24] who models a single traffic intersection and solves the optimal control problem associated with it. A technical enhancement to the type of model used by De Schutter is adapted from the work of Ban et al [25]. We also use the idea proposed by Keshavarz et al. in [23] when solving the inverse optimal control problem for a traffic controller due to its computational efficiency. In other words, we formulate and solve a single optimization problem to solve the inverse optimal control problem instead of the repeated solving of several optimal control problems as seen in methods proposed previously.
3.1.3 Outline and contributions

One of the main contributions of this thesis is the development of a computationally efficient method to solve the inverse optimal control problem for a traffic controller at an intersection via convex programming, which we introduced in [10], and an exploration of non-convex programming in one instance of the problem. This chapter extends our preliminary work by addressing several challenges of applying the method on experimental data. We extend the functionality to work with fixed time and real time control strategies under certain conditions. We explain how concepts of approximately optimal and infeasibility can be used to assess the performance of the method on experimental data, without knowledge of the true solution. Finally, supporting source code has been improved to exploit faster numerical solvers.

We first review the relevant components of the single intersection optimal control problem in Section 3.2. In Section 3.3, we build the objective function in terms of a basis of features with unknown weights and derive the necessary and sufficient conditions for optimality of the optimal control problem. We then use the optimality conditions as constraints in the setup of the inverse optimal control problem. In Section 3.4, we discuss the feasibility of solutions to the inverse optimal control problem and the related notion of approximate optimality and finally, in Section 3.5, numerical and field experiments are performed on different intersections and controllers to test the performance of the proposed method.

3.2 Intersection traffic model and optimal signal control

We review a model of traffic flow at a single intersection, and describe an optimal control problem to compute switching times of the traffic signal controller, originally proposed in [24]. The model and the optimal control problem are essential elements needed to build the inverse optimal control problem in Section 3.3. A detailed analysis of the model and optimal control extensions can be found in [24]; we repeat only the relevant details here for completeness.
3.2.1 Continuous time dynamics of traffic flow at an intersection

Consider a model of a single intersection with \( m \) links indexed by \( i \), managed by a traffic signal controller. The queue lengths \( q_i(t) \) evolve in continuous time \( t \), and \( k \) denotes the number of phase switches observed since the initial time \( t_0 \). The queue lengths \( q_i(t) \) on each incoming link \( i \) at time \( t \) evolve according the following first order linear hybrid system:

\[
\frac{dq_i(t)}{dt} = s_i(t) + a_{i,k} - \mu_{i,k} \\
0 \leq s_i(t) \perp q_i(t) \geq 0,
\]

where \( \perp \) indicates complementarity of \( s_i(t) \) and \( q_i(t) \), \( s_i(t) \) is a slack variable for the queue on link \( i \), \( a_{i,k} \geq 0 \) denotes the arrival rate on link \( i \) during phase \( k \), and \( \mu_{i,k} \geq 0 \) denotes the saturation rate on link \( i \) during phase \( k \). We assume the saturation rate is zero when the light is red for link \( i \) during phase \( k \).

In the model (3.1), the queue length function \( q_i(t) \) is piecewise affine within a phase \( k \). If \( t_k \) denotes the switching time when the \( k^{th} \) phase ends, the lengths of the queues at the switching times can be computed as

\[
q_i(t_{k+1}) = \max \left\{ \left( a_{i,k+1} - \mu_{i,k+1} \right) \delta_{k+1} + q_i(t_k), 0 \right\},
\]

where \( \delta_k = t_k - t_{k-1} \) is the duration of the \( k^{th} \) phase. To simplify our notation moving forward, let \( \alpha_{i,k} = a_{i,k} - \mu_{i,k} \) denote the net flow into link \( i \) during phase \( k \). The variable \( q_{i,k} = q_i(t_k) \) denotes the queue length at time \( t_k \) on link \( i \), and \( q_k = [q_{1,k}, \cdots, q_{m,k}]^T \) is the vector of queue lengths at time \( t_k \).

3.2.2 Optimal signal control via switching time control

With the model of the intersection defined, the finite horizon optimal control problem of minimizing some objective function \( \tilde{J} \) over the \( m \times (n + 1) \) state variables \( q = [q_0^T, \cdots, q_n^T]^T \) and the \( n \) control variables \( \delta = [\delta_1, \cdots, \delta_n]^T \) can be written as the following extended linear complementarity problem (ELCP):

16
minimize \( q, \delta \):

\[
\tilde{J}(q, \delta)
\]

subject to:

\[
q_{i,k+1} \geq \alpha_{i,k+1} \delta_{k+1} + q_{i,k} \quad \forall i \in \{1, \ldots, m\}, \\
q_{i,k} \geq 0 \quad \forall i \in \{1, \ldots, m\}, \\
q_{i,0} = q_{i0} \quad \forall i \in \{1, \ldots, m\} \\
\delta_k \geq \delta_{\text{min}} \quad \forall k \in \{1, \ldots, n\} \\
\delta_k \leq \delta_{\text{max}} \quad \forall k \in \{1, \ldots, n\} \\
\sum_{k=1}^{n} \delta_k = t_f - t_0.
\]

where \( t \in [t_0, t_f] \) is given in terms of the initial time \( t_0 \) and the final time \( t_f \).

The parameters \( \delta_{\text{min}} \) and \( \delta_{\text{max}} \) are the upper and lower bounds of the phase duration, and prevent the signal from switching too quickly or not at all. Note that (3.3) is nonlinear due to the following constraint:

\[
(q_{i,k+1} - \alpha_{i,k+1} \delta_{k+1} - q_{i,k}) q_{i,k+1} = 0, \\
\forall i \in \{1, \ldots, m\}, \forall k \in \{0, \ldots, n-1\}
\]

which requires that either \( q_{i,k+1} - \alpha_{i,k+1} \delta_{k+1} + q_{i,k} \) or \( q_i(t_{k+1}) \) is equal to zero.

As identified in [24], problem (3.3) can be relaxed to a linear constraint set by dropping the complementarity constraints, yielding:

minimize \( q, \delta \):

\[
\tilde{J}(q, \delta)
\]

subject to:
\begin{align*}
q_{i,k+1} \geq \alpha_{i,k+1} \delta_{k+1} + q_{i,k} & \quad \forall i \in \{1, \cdots, m\}, \\
& \forall k \in \{0, \cdots, n-1\} \\
q_{i,k} \geq 0 & \quad \forall i \in \{1, \cdots, m\}, \\
& \forall k \in \{1, \cdots, n\} \\
q_{i,0} = q_{i0} & \quad \forall i \in \{1, \cdots, m\} \\
\delta_k \geq \delta_{\min} & \quad \forall k \in \{1, \cdots, n\} \\
\delta_k \leq \delta_{\max} & \quad \forall k \in \{1, \cdots, n\} \\
\sum_{k=1}^n \delta_k = t_f - t_0 &
\end{align*}

(3.4)

Note that in (3.4), the constraint set is linear, and therefore it can be written as:

\[
\text{minimize}_{x \in \mathcal{X}} : J(x)
\]

where

\[
x = \begin{bmatrix} q \\ \delta \end{bmatrix},
\]

and

\[
\mathcal{X} = \{ \tilde{x} \in \mathbb{R}^{(n+1)m+n} \mid A\tilde{x} \leq b, A_{eq}\tilde{x} = b_{eq} \},
\]

for suitable \( A, A_{eq}, b, \) and \( b_{eq}. \)

An important result from [24] links problem (3.4) and (3.3). Specifically, if \( \tilde{J}(q, \delta) \) is a strictly increasing function of the queue lengths \( q, \) [24] showed the optimal solution of (3.4) is also optimal for (3.3). Thus, for a restricted class of objective functions, the ELCP optimal control problem (3.3) can be solved with a tight convex relaxation.

### 3.3 Recovering switching control objectives via inverse optimal control

#### 3.3.1 Inverse optimal control definition

The inverse optimal traffic signal problem can be stated as follows. Given an observation of the system trajectory \( x^* = \begin{bmatrix} q^* \\ \delta^* \end{bmatrix}, \) find the weights \( w^* \in \mathcal{W} \)
such that

\[ x^* = \arg\min_{x \in \mathcal{X}} \sum_{j=1}^{n_f} w_j^* J_j(x), \]  

(3.5)

where \( J_j(x), j \in \{1, \cdots, n_f\} \) represents the \( n_f \) features of the objective function, or equivalently, the basis of the objective function. (The objective function \( J(x) \) is a linear combination of the features and the weights \( w_j^* \) where \( j \in \{1, \cdots, n_f\} \).) In other words, the system trajectory should be optimal for (3.5) under some weights \( w^* \), which are to be estimated in the inverse optimal control problem. Since the system trajectory \( x^* \) is assumed to be optimal, it must satisfy the Karush–Kuhn–Tucker (KKT) conditions for optimality. The conditions are given by:

\[
\begin{align*}
\sum_{j=1}^{n_f} w_j \nabla J_j(x) + A^T \lambda + A_{eq}^T \nu &= 0 \\
A_{eq} x - b_{eq} &= 0 \\
Ax - b &\leq 0 \\
\lambda^T (Ax - b) &= 0 \\
\lambda &\geq 0,
\end{align*}
\]

(3.6)

where \( \lambda \) and \( \nu \) are the Lagrange multipliers associated with the inequality and equality constraints respectively.

If \( \delta_{\min} < \delta_{\max} \), the KKT conditions (3.6) are also sufficient conditions for (3.5) because Slater’s constraint qualification holds [26]. This follows from the fact that an interior point \( \hat{x} = \begin{bmatrix} \hat{q} \\ \hat{\delta} \end{bmatrix} \) can be easily constructed by selecting some \( \delta_{\min} < \hat{\delta}_k < \delta_{\max} \) for all \( k \), such that the phase durations \( \hat{\delta} \) are strictly feasible. Then, for each link \( i \), we initialize the queue lengths according to the initial data \( \hat{q}_{i,0} = q_{i,0} \). Now, the strictly feasible queue lengths can be computed according to \( \hat{q}_{i,k+1} = \max\left\{ \alpha_{i,k+1} \hat{\delta}_{k+1} + \hat{q}_{i,k}, 0 \right\} + \varepsilon \), for some \( \varepsilon > 0 \). Because there are no upper bounds on the queue lengths, the evolution of the queues given by \( \hat{q} \) is strictly feasible when computed in this way.

### 3.3.2 Convex inverse optimal control problem

Given the necessary and sufficient conditions for optimality, we can now use them to solve the inverse optimal control problem. Following the generalized inverse optimal control approach proposed in [23], we treat the KKT conditions
for optimality as constraints to the inverse optimal control problem. This
approach guarantees that the weights returned by the following optimization
problem are indeed the weights that satisfy the optimal control problem (3.5).
Moreover, the optimization is itself a convex program, which can be solved
without repeatedly solving the optimal control problem, common to many
approaches.

The *approximately optimal* [23] inverse optimal control program is:

$$\text{minimize}_{r,\lambda,\nu,w} :$$

$$\sum_{l=1}^{2} \|r_l\|_2^2$$

subject to :

$$\sum_{j=1}^{n_f} w_j \nabla J_j(x^*) + A^T \lambda + A_{eq}^T \nu = r_1$$

$$\lambda^T (Ax^* - b) = r_2$$

$$\lambda \geq 0$$

$$w \geq 0$$

$$\sum_{j=1}^{n_f} w_j = 1$$

(3.7)

In (3.7) above, the decision variables are the weights $w$, the Lagrange vari-
able $\lambda$ and $\nu$, and the residuals $r_l$. The objective minimizes the sum of the
squares of the 2–norm of the residuals, and takes the value zero when the KKT
conditions are exactly satisfied. In general, the conditions need not be satis-
fied, especially if there are errors in the model used within the inverse optimal
control problem (e.g. the objective basis functionals do not span the space of
the true control objective, the incoming and outgoing flows have some error,
etc.). Instead, the KKT conditions are allowed to be approximately satisfied,
which yields an approximately optimal inverse optimal control problem [23].
Also note that in this problem, the objective function of the optimal control
problem $J(x)$ must be strictly increasing with respect to the queue length in
order for the KKT conditions to be both necessary and sufficient for optimal-
ity. Therefore, we add additional constraints in the implementation of the
inverse optimal control problem to ensure that this holds. The details of the
features of the objective function that are increasing with respect to queue
length are explained in Section 3.3.3.
3.3.3 Convex features of the objective function

The purpose of the inverse optimal control method is to estimate a traffic controller’s signal control strategy in order to reliably predict the controls at an intersection for a given in-flow of traffic into an intersection. Therefore, the definition of the control strategy must be rich enough to explicitly produce a set of controls given a trajectory of inputs. We define a traffic controller’s strategy in terms of an optimization problem with an objective function that is comprised of a linear combination of features. These features are functions of the state that are reasonable objectives for a traffic controller to optimize over. For example, one could easily imagine that a traffic controller would attempt to minimize the queue lengths on all sides of the intersection. Since the objective function is a linear combination of features, we hope to recover some linear combination that is equivalent to the observed traffic control strategy. A summary of features used in our work is shown in Table 3.1.

The first feature, the variance of the cycle length, is a reasonable feature for a traffic controller to minimize since most pre-timed signals and some actuated signals maintain a constant cycle length in their control strategy. The variable $c_{\text{max}}$ is defined as the total number of cycles in the time-horizon under consideration, $n_c$ is defined as the number of traffic signal phases in one cycle and $\bar{\delta}_c$ is the average cycle length over all cycles in the time horizon. In fact, since $\bar{\delta}_c$ is always fixed for a given time-horizon $t_f$ and a given maximum number of cycles $c_{\text{max}}$, this feature has the same minimizer as

$$
\sum_{c=1}^{c_{\text{max}}} \left( \sum_{p=1}^{n_c} \delta_{n_c(c-1)+p} \right)^2.
$$

(3.8)

The next set of features compute the variance of the phase lengths of eight different traffic signal phases that we have observed in our field tests:

$$
\sum_{c=1}^{c_{\text{max}}} \left( \delta_{n_c(c-1)+p} - \bar{\delta}_p \right)^2, \quad p \in \{1, 2, \ldots, n_c\}.
$$

(3.9)

Here, $\bar{\delta}_p$ is the average time length of all phases in the time-horizon that were the $p^{th}$ traffic signal phase. These features are useful to describe the occurrence of a phase that always exists for a fixed amount of time. Most pre-timed signals
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance of cycle length</td>
<td>$\sum_{c=1}^{c_{\text{max}}} \left( \sum_{p=1}^{n_c} \delta_{n_c(c-1)+p} - \bar{\delta}^c \right)^2$</td>
</tr>
<tr>
<td>Variance of phase length</td>
<td>$\sum_{c=1}^{c_{\text{max}}} \left( \delta_{n_c(c-1)+p} - \bar{\delta}^p \right)^2, p \in {1, 2, \cdots, n_c}$</td>
</tr>
<tr>
<td>Queue lengths</td>
<td>$\sum_{k=0}^{n} q_{i,k}, i \in {1, 2, \cdots, m}$</td>
</tr>
<tr>
<td>Squares of queue lengths</td>
<td>$\sum_{k=0}^{n} q_{i,k}^2, i \in {1, 2, \cdots, m}$</td>
</tr>
<tr>
<td>Phase length error</td>
<td>$\sum_{c=1}^{c_{\text{max}}} \left( \delta_{n_c(c-1)+p} - \bar{\Delta}^p \right)^2, p \in {1, 2, \cdots, n_c}$</td>
</tr>
</tbody>
</table>

Table 3.1: Numbered list of features.

and some actuated signals have phases that exhibit this behavior.

Similarly, we include another set of features that compute the phase length error of each traffic signal phase as follows:

$$\sum_{c=1}^{c_{\text{max}}} \left( \delta_{n_c(c-1)+p} - \bar{\Delta}^p \right)^2, p \in \{1, 2, \cdots, n_c\},$$

(3.10)

where $\bar{\Delta}^p$ is the average phase length of all phases in the observed controls of the traffic controller that were the $p^{th}$ traffic signal phase. These features serve the same purpose as the set of features described by (3.9), but they include a background term ($\bar{\Delta}^p$) that penalizes any deviation from the observed phase lengths.

Additionally, we use two more sets of features that compute the sum of the queue lengths (3.11) and the sum of the squares of the queue lengths (3.12) at the phase switching times on each queue at the intersection respectively.

$$\sum_{k=0}^{n} q_{i,k}, i \in \{1, 2, \cdots, m\}$$

(3.11)

$$\sum_{k=0}^{n} q_{i,k}^2, i \in \{1, 2, \cdots, m\}$$

(3.12)

The difference between the two sets of features is the way they penalize a set of queue lengths.
3.4 Discussion of feasibility of solutions and approximate optimality

3.4.1 Feasibility

Note that in the inverse optimal control problem, the sum of the squares of the 2-norm of the residuals of the KKT conditions ($r_1$ and $r_2$) is minimized. This clearly does not guarantee that $r_1 = 0$ and $r_2 = 0$ and therefore the necessary and sufficient conditions for optimality might not always be strictly satisfied. However, also note that the complete set of KKT conditions are not included as constraints in the IOC problem (3.7). In fact, the full set of KKT conditions as residuals is given by:

\[
\begin{align*}
    r_1 &= \sum_{j=1}^{n_f} w_j \nabla J_j(x) + A^T \lambda + A_{eq}^T \nu \\
    r_2 &= \lambda^T (Ax - b) \\
    r_3 &= A_{eq}x - b_{eq} \\
    r_4 &= \max\{0, Ax - b\},
\end{align*}
\]  

(3.13)

where the max operator is defined element-wise. Note that we do not include the primal feasibility constraints associated with the residuals $r_3$ and $r_4$ in the IOC problem because they are not functions of the Lagrange variables $\lambda$ and $\nu$.

Therefore, satisfaction of the KKT conditions depends on $x$, the observed state trajectory of the true system, and the accuracy of the mass conservation model, embedded in the linear equality and inequality constraints. For the purposes of discussion, we shall refer to the observed state trajectory $x$ as simply the observed trajectory and similarly, the state trajectory obtained by solving the optimal control problem with the objective function recovered by solving the IOC problem will be referred to as the simulated trajectory.

Now, if the KKT conditions are not strictly satisfied, i.e. there is violation of the residuals, the solution of the inverse optimal control problem is not guaranteed to be feasible. This violation can occur if the solution to the IOC problem produces residuals $r_1 \neq 0$ and $r_2 \neq 0$ or if there is error in the traffic model or the observed trajectory data, in which case $r_3 \neq 0$ and $r_4 \neq 0$. More precisely, if the KKT conditions are violated by the model of traffic used in
the formulation of the problem, the solution to the IOC problem (3.7) does not guarantee an objective function under which the observed trajectory is optimal. Therefore, it is important to check the residual values and verify whether the problem is feasible or approximately feasible.

Significant violations of the residuals imply that either the model or the data has significant error and as a result, the output of the IOC method is not reliable. However, the idea is that if the violation is not significant, we might still be able to recover an objective function under which the observed trajectory is close to optimal or \textit{approximately optimal}. In other words, we might be able to recover an objective function that produces a simulated trajectory that is similar to the observed trajectory. There is a notion here that a solution that is close to the optimal solution in terms of objective value will be similar to the optimal solution in the solution space. This notion is discussed further in Section 3.4.2.

The significance of violation of the residuals to an IOC problem is something that depends on each individual problem and therefore, we construct performance metrics in Section 3.5 to evaluate the violation of the KKT conditions in a problem and whether we have in fact recovered an objective function that is able to mimic the behavior of the true system.

However, in some situations, we can definitively gauge the quality of a solution by examining the violations of the KKT conditions. For example, suppose the violation of the residuals \( r_3 \) and \( r_4 \) is small (i.e. our model and data fit each other well), the magnitude of the vector \( r_1 \) is a measure of how good the selected feature set is for a particular problem. A large magnitude of \( r_1 \) indicates that the feature set is not rich enough to recover the observed control strategy. This follows from the fact that the model and the data do not contain much error, but an objective function that describes the behavior of the controller reliably was not found.

3.4.2 Approximate Optimality

An approximately optimal solution is defined in [23] as one where the residuals \( r_1, r_2, r_3, \) and \( r_4 \) are close to zero. The hope with the approximately optimal solution is that since the residuals are close to zero and the problem is convex,
the optimal solution is near the approximately optimal solution and that they are similar. This, of course, is not guaranteed and thus, it is possible to have an approximately optimal solution that is very different from the exactly optimal solution. However, this case can be easily checked when the true optimal solution is known.

However in practice, we do not generally have access to the exact objective function over which the true system is optimizing and in these cases, heuristics could be used to evaluate the quality of the approximately optimal solution. For example, one way of generating a heuristic is to solve similar problems where the exact objective is known and set a threshold for each of the residuals’ violation based on the results. The heuristic could state that if the solution to a problem produces residuals that breach a threshold, that solution is of a lower quality than ones that do not breach the threshold.

Another way to evaluate the quality of approximately optimal solutions to the IOC problem is to solve the associated optimal control problem with the recovered objective and compare the observed and simulated trajectories. Since our immediate goal is to produce a simulated trajectory that mimics the observed trajectory, this is the approach taken to evaluate the performance of the experiments described in this chapter.

3.5 Experiments

This section tests the performance of the inverse optimal control method in estimating the control objective through several experiments and performance metrics. The experiments are performed on two layouts of intersections, both of which are intersections of two two-way streets.

3.5.1 Performance metrics

Several performance metrics are defined to assess the accuracy of the inverse optimal control solver. The absolute percent error on the objective is computed in the following way:

\[
e_{\text{obj}} = 100 \times \frac{\| \sum_j w_j J_j (x^*) - \sum_j w_j J_j (\tilde{x}) \|_1}{\sum_j w_j J_j (x^*)},
\]  

(3.14)
where \( w_j \) are the weights estimated by the inverse optimal control problem, \( x^* \) is the true system’s trajectory, and \( \tilde{x} \) is the estimator’s trajectory. The estimator’s trajectory is obtained by solving the optimal control problem 3.4 with the objective recovered from the inverse optimal control problem. Therefore, \( \sum_j w_j J_j(x^*) \) and \( \sum_j w_j J_j(\tilde{x}) \) represent the objective values of the true system’s trajectory and the estimated system’s trajectory with respect to the recovered control objective. In other words, this metric measures the optimality gap between the observed and simulated trajectories under the learned feature weights. The metric is defined in this way because practically, we will not have access to the true system’s control objective and therefore, we cannot compare the recovered objective with the true system’s objective.

The next performance metric we define is the absolute percent error of the phase lengths within the time horizon:

\[
e_\delta = 100 \times \frac{\|\delta^* - \tilde{\delta}\|_1}{t_f}
\] (3.15)

where \( \delta^* \) and \( \tilde{\delta} \) are vectors of phase lengths of the true system and the estimated system respectively. This metric quantifies the significance of the error in the phase lengths with respect to a given time-horizon.

The next metric we define is \( e_o \), the optimality error,

\[
e_o = \frac{1}{n_x + 1} \sum_{l=1}^2 \|r_l\|_2^2.
\] (3.16)

Note that, this is equivalent to the objective function of the inverse optimal control problem divided by the sum of the number of elements in the residual vectors \( r_1 \) and \( r_2 \). \( n_x \) is the number of elements in the state vector \( \tilde{x} \) and consequently in \( r_1 \), and the number of elements in \( r_2 \) is always one. We choose this metric because it expresses the magnitude of the average violation of residuals. We assume here that a small residual means that we are most likely close to the optimal solution, although, there is no guarantee. In fact, the obtained solution might not even be strictly feasible as described in Section 3.4. We quantify the level of feasibility of the solution as follows:
\[ e_i = \frac{1}{n_\lambda + n_\nu} \sum_{i=3}^{4} \|r_i\|_2^2, \]  

(3.17)

where \( r_3 \) and \( r_4 \) are the residuals of the KKT conditions that are not functions of the dual variables \( \lambda \) and \( \nu \). Again, we divide the sum of the absolute values of the residuals by the number of elements in the residuals \( r_3 \) and \( r_4 \). \( n_\lambda \) and \( n_\nu \) are the number of elements in the vectors \( \lambda \) and \( \nu \) respectively. We refer to this metric as the \textit{infeasibility error}. Ideally, \( e_i \) should be identically zero, however, in reality, due to errors in the data and errors in the model, a small \( e_i \) is expected.

We will now use these performance metrics to test the inverse optimal control method and analyze the results of empirical experiments in the following sections.

### 3.5.2 Field experiments

In the following experiments, we consider two intersections of two two-way streets. In one instance of the problem, we consider an intersection that has no dedicated left-turn lanes (Figure 3.1a). The traffic controller for this intersection switches between phases 1 and 2 from Figure 3.2 only. In another instance of the problem, we consider an intersection with dedicated left-turn lanes on all streets (Figure 3.1b) and assume that the traffic controller has access to all eight phases shown in Figure 3.2.

First, we test the IOC method with experimental data collected with the help of TrafficTurk. We solve the inverse optimal control problem to estimate the unknown weights of the objective function and then use the estimated weights to simulate the controller by solving the optimal control problem. We call the switching policy recorded by TrafficTurk the \textit{true system trajectory} or the \textit{observed trajectory}; the policy generated with the estimated weights is called the \textit{estimator trajectory} or the \textit{simulated trajectory}.

#### 3.5.2.1 Experimental setup of field experiments

Field experiments use experimental data collected with TrafficTurk and knowledge of the traffic signal phases that occurred while collecting data to estimate
Figure 3.1: Intersection layouts used in numerical and empirical experiments.

Figure 3.2: The 8 traffic signal phases considered in the intersection layout shown in 3.1b. For the intersection layout in 3.1a, we only consider phases 1 and 2.
the traffic controller strategy of an intersection. We first solve the inverse optimal control problem and then use the recovered objective function to solve the optimal control problem to obtain a simulated trajectory and compare it with the observed trajectory.

**Data collection** Empirical data was collected using a preliminary version of TrafficTurk on a Samsung GT-I9103 device. A turning movement count is collected by observing a vehicle perform a maneuver at an intersection and swiping the maneuver on a graphic of an intersection on the screen of the smart-phone.

Turning movement counts were collected at two intersections; one of them used a pre-timed traffic controller and the other used an actuated traffic controller. Before collecting data, the traffic controllers were observed until all the different traffic signal phases used in the intersection were seen. This was necessary since the inverse optimal control method requires knowledge of the traffic signal phases.

Then, each vehicle that passed the intersection was recorded with TrafficTurk. In addition, the times of the phase switches were recorded on another phone since the state trajectory is comprised of the time lengths of each phase as well. When fully implemented, collection of phase timing data will not be required since traffic controller strategy estimation will be supported by a *Hidden Markov Model* based algorithm to estimate phases of each recorded maneuver [9].

In this particular experiment however, the two sets of data (turning movement counts and times of phase switches) were merged to create a list of maneuvers and phase switches as seen in an example in Table 3.2. A \texttt{maneuver\_id} value of -1 indicates that a phase switch occurred at the corresponding \texttt{timestamp} value. Any other values of \texttt{maneuver\_id} denotes a valid maneuver that was observed on the intersection. The \texttt{timestamp} value is in units of milliseconds since 00:00hrs January 1, 1970 GMT. From this raw data, an observed trajectory \((x)\) can be constructed. The phase lengths in the trajectory can be obtained directly from the times of the phase switches.

In order to estimate the queue lengths at the times of the phase switches, we first calculate the average arrival and departure rates of vehicles in each
<table>
<thead>
<tr>
<th>node_id</th>
<th>maneuver_id</th>
<th>timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>7307</td>
<td>−1</td>
<td>1368905758095</td>
</tr>
<tr>
<td>7307</td>
<td>68492</td>
<td>1368905759179</td>
</tr>
<tr>
<td>7307</td>
<td>36289</td>
<td>1368905761314</td>
</tr>
<tr>
<td>7307</td>
<td>68492</td>
<td>1368905762420</td>
</tr>
<tr>
<td>7307</td>
<td>36289</td>
<td>1368905763621</td>
</tr>
<tr>
<td>7307</td>
<td>68492</td>
<td>1368905765534</td>
</tr>
<tr>
<td>7307</td>
<td>36289</td>
<td>1368905766574</td>
</tr>
<tr>
<td>7307</td>
<td>68492</td>
<td>1368905767996</td>
</tr>
<tr>
<td>7307</td>
<td>68511</td>
<td>1368905769542</td>
</tr>
<tr>
<td>7307</td>
<td>68483</td>
<td>1368905772812</td>
</tr>
<tr>
<td>7307</td>
<td>−1</td>
<td>1368905772812</td>
</tr>
</tbody>
</table>

Table 3.2: A representation of the format of data collected by TrafficTurk.

phase. We assume that the arrival rate is constant throughout the period of time when data was collected and therefore the arrival rate for each phase on each link can be approximated as,

\[
a_{i,k} = \frac{\# \text{ maneuvers that started from link } i}{\text{time period of data collection}}. \tag{3.18}
\]

The departure rates can be exactly calculated for each phase and each link by,

\[
\mu_{i,k} = \frac{\# \text{ maneuvers that started from link } i \text{ during phase } k}{\text{time period of the } k^{th} \text{ phase}}. \tag{3.19}
\]

Other parameters such as \( t_0, t_f, m, n, c_{max}, \) and \( n_c \) can also be easily deduced from the turning movement counts. With these parameters, the observed phase lengths and a reasonable assumption for \( q_{i,0} \), we can calculate the queue lengths on each link at the phase switching times using (3.2).

### 3.5.2.2 Pre-timed signal

Pre-timed signals are the most basic type of traffic controller and also the most common [11]. Therefore, it is meaningful for the inverse optimal control estimator to be able to estimate the strategies of pre-timed traffic controllers. Pre-timed traffic controllers operate by having a fixed amount of time in each
phase and a fixed phase sequence in each cycle. The specific traffic signal controller that we test in this experiment is the one present at the intersection of First St. and Green St. in Champaign, IL.

The intersection is a four-way intersection with the layout shown in Figure 3.1a. TrafficTurk data was collected here along with the times of the phase switches. The traffic controller at the intersection only uses phases 1 and 2 in a cycle as illustrated in Figure 3.2.

We use a convex feature set for this problem since we have explicitly designed convex features that describe the objectives of a pre-timed signal, i.e. minimizing the variance of the individual phase lengths and minimizing the variance of the cycle lengths. The IOC method is able to mimic the controller well (as shown in Figures 3.3 and 3.4) with a relatively small absolute percent error on the phase lengths \( e_\delta = 12.10\% \) and a small absolute percent error on the objective \( e_{\text{obj}} = 0.038\% \). The interpretation of this result is that we have found an equivalent, but slightly different policy under the chosen feature set. Moreover, the infeasibility error is small \( e_i = 4.58 \times 10^{-16} \) meaning that the collected data fits our model well. The optimality error is also small \( e_o = 1.20 \times 10^{-4} \) meaning that the observed trajectory is indeed close to an optimal solution of (3.5) under the recovered objective function, i.e. approximately optimal.

With a pre-timed traffic signal, we would expect the recovered weights of the objective function to be favored towards the features that minimize the variance of the cycle length and each of the phase lengths.

However, this is not the case. The recovered weights are favored towards the features that minimize the queue lengths at the switching times. The reason for this is that there could be multiple objective functions under which the observed policy is optimal for the given traffic flow. For example, the control strategy that minimizes the variance of the phase lengths in a certain way could also be one that minimizes the queue lengths at the switching times. One way to reduce this non-uniqueness would be to solve the inverse optimal control problem with data from multiple days and multiple states of traffic simultaneously.

Another interpretation of these non-unique weights is that the traffic controller is in fact well timed to reduce the queue lengths at the switching times.
### Observed vs. Simulated phase lengths

<table>
<thead>
<tr>
<th>Phase #</th>
<th>Observed phase lengths</th>
<th>Simulated phase lengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.3: Observed and simulated phase lengths of a pre-timed controller.

for the prevalent traffic conditions. Note that in most cases, we are looking for an objective function that mimics the controller well so that we may predict or estimate the controller’s controls at a future time and hence an equivalent but different objective function than the true objective function is still an acceptable result. However, the accuracy of the estimation required depends on the specific application.

#### 3.5.2.3 Actuated signal

The other common type of traffic controller on urban surface streets is the actuated signal [27]. Actuated signals use sensors (inductive loops, cameras etc.) to detect vehicles in the vicinity of an intersection. Then, the controllers decide the phase length of each phase (some phases can be skipped entirely). An actuated signal located at the intersection of First St. and Springfield Ave. in Champaign, IL was used to test the inverse optimal control estimator on actuated signals with a convex set of features.

The layout of the intersection is shown in Figure 3.1b. Since the model that we use assumes that the phase sequence within a cycle remains constant for all cycles, we include all of the 8 phases from Figure 3.2 in every cycle but allow the phases that are sometimes skipped by the controller to have a minimum phase length constraint of zero. From our observations, it was evident that certain phases would always be part of a cycle, whereas other
Figure 3.4: Observed and simulated phase switches of a pre-timed controller.

phases would only be a part of the cycle if certain traffic conditions prevailed. More specifically, phases 1 and 2 were part of every cycle whereas one of phases 3, 5, and 6 preluded phase 1 and one of phases 4, 7, and 8 preluded phase 2 sometimes.

The results of the estimator suggest that this type of signal does not mimic the observed traffic controller very well, as it can be seen in Figures 3.5 and 3.6. Some phases that were present in the observed trajectory do not appear in the simulated trajectory and the phase lengths of the simulated trajectory do not match the phase lengths of the observed trajectory. Thus, the absolute percent error of the phase lengths was $e_\delta = 56.77\%$. However, the infeasibility error ($e_i = 5.99 \times 10^{-18}$), absolute percent error on the objective ($e_{obj} = 1.04\%$) and the optimality error ($e_o = 2.56 \times 10^{-6}$) are small. This suggests that the collected data fits the model well, the observed and simulated policies are similar under the recovered objective function and that the solution is approximately optimal respectively. But, the fact that the simulated policy is so different from the observed policy lets us infer that the objective function’s basis (our chosen feature set) is not rich enough to accurately describe the actuated signal’s strategy as an optimization problem. Therefore, in the next section, we explore the possibility of using richer feature sets that are non-convex in order to recover control strategies of different kinds of controllers.

One of the issues with the actuated signal is that the control logic that it
Figure 3.5: Observed and simulated phase lengths for an actuated signal.

Figure 3.6: Observed and simulated switching times for an actuated signal.
uses is based on a binary variable that relates to the existence of vehicles on one or more streets of an intersection. Thus, developing features that capture this logic using binary variables is not possible under the current framework. This is due to the fact that the KKT conditions hold for objective functions and constraints that are continuously differentiable. This thesis focuses on the merits and limitations of the current framework using continuously differentiable objective functions and constraints in order to check the feasibility of the framework for the current application.

3.5.3 Experiments with non-convex feature sets

So far, we have only dealt with convex feature sets so as to keep the inverse optimal control problem convex and so that the KKT conditions are necessary and sufficient for the associated optimal control problem. While this ensures that the problem can be solved efficiently, it also limits the richness of different control strategies that can be recovered. This is problematic if, for instance, a human traffic controller’s behavior is optimal under a feature set that is non-convex, and no linear combination of convex features is able to fully capture the behavior.

Therefore, we introduce a common non-convex feature that is frequently mentioned in literature on optimal control of traffic intersections - the delay minimizer. This feature is also one that is a reasonable one for a human traffic controller to optimize over even though they might not be explicitly solving the associated optimal control problem. In other words, the actions of a human controller could be equivalent to minimizing delay for vehicles at the intersection.

The feature is defined as:

$$J_{\text{delay}}(x, t) = \int_{t_0}^{t_f} q_i(t) dt, \ i \in \{1, \cdots, m\}. \quad (3.20)$$

A similar feature is used in [24] that has an equivalent minimizer to (3.20). In order to use this feature within the framework of our problem, i.e. as a function of queue lengths at the phase switching times and the phase lengths, we use a simplified version (3.21) that was also proposed by De Schutter in
(3.21) is equivalent to (3.20) if there is no lower bound saturation of the queue lengths within any of the phases, otherwise, it is still a reasonable approximation [24]. This feature is also equivalent to the sum of the uniform delay and over-saturated delay at the intersection as defined in [28].

\[ J_i^{delay}(q, \delta) = \sum_{k=1}^{n-1} \delta_k(q_{i,k} + q_{i,k+1}), \ i \in \{1, \cdots, m\} \] (3.21)

Now, we test the performance of the inverse optimal control method when a non-convex objective function is used to generate the observed trajectory. We first test the inverse optimal control method when the estimator contains the same non-convex feature set as the true system and then, we also test the situation where the estimator only contains convex features. For the purposes of these experiments, we were not able to identify any traffic controllers in our area that ran under a delay-minimizing objective function. Therefore, we use VisSim [29], a traffic simulation software package to build and simulate an intersection under the delay minimizing objective. The exact procedure for running this experiment is provided in the following section.

3.5.3.1 Procedure for implementing the delay minimizing signal with VisSim

VisSim is a microscopic traffic flow simulation software package [29]. We use VisSim to test a delay minimizing signal because we did not have access to a controller of this type in the field. VisSim is used to mimic the traffic flow into and out of an intersection. It provides us with a way to avoid the inverse crime setting while testing the inverse optimal control method. We first build an intersection in VisSim that is similar to the intersection shown in Figure 3.1a. The different routes that vehicles can take in the intersection are defined and each route has a thin vehicle detector placed on it. The traffic controller is defined as having two phases in a cycle. (phase 1 and 2 in Figure 3.2).

Then, the signal timing plan is created. We define the signal plan as having multiple cycles. We use the first half of these cycles to warm-up the simulation and the rest to implement the delay minimizing signal plan. The delay minimizing signal plan is obtained by solving the optimal control problem with a pre-specified objective function. In the case of experiments with the
delay minimizing feature, we set all the weights in the objective function to be zero except for the weights associated with the cycle length variance feature and the delay features ($J_i^{delay}$). We set all of the the non-zero weights to be equivalent to each other (these could be randomly drawn instead). We then use a non-linear solver to solve the optimal control problem multiple times and choose the solution that produces the smallest objective value.

Using the phase switching times produced by the optimal control solver, we complete defining the signal plan in VisSim. We then setup the traffic in-flows in VisSim to have the same parameters as the ones used in the optimal control problem and select the “Default” option for the rest of the traffic parameters. By running this signal plan and extracting the vehicle detector data (available at every second of the simulated time period), we can then infer the vehicle maneuvers and their approximate times. We use a script written in the Python scripting language to export the data back to the inverse optimal control solver in the same format as TrafficTurk data.

3.5.3.2 Results

The first attempt at recovering the observed trajectory generated by a VisSim simulation of a delay-minimizing traffic controller uses a feature set consisting of all features from Table 3.1 and $J_i^{delay}$. Since the delay feature, and as a result the optimal control problem, are both non-convex, the KKT conditions are no longer sufficient conditions for the optimal control problem. Therefore, we solve the convex IOC problem once and then solve the optimal control problem multiple times with different starting points given to a non-linear solver and choose the solution that gives the smallest objective value. In this particular case, we solved the problem 100 times and chose the best solution. Note that the solution to the convex IOC problem is guaranteed to be globally approximately optimal, but there is no guarantee on the unique-ness of the solution. The solution to the optimal control problem, on the other hand, is locally approximately optimal.

The estimator was able to mimic the traffic controller’s switching policy well ($e_\delta = 6.72\%$) and this is seen in Figures 3.7a and 3.8a. The infeasibility error and the optimality error were small ($e_i = 8.79 \times 10^{-18}$, $e_o = 4.11 \times 10^{-11}\%$). The absolute percent error in the objective was also relatively small ($e_{obj} =
7.36%). Since the performance metrics related to optimality and infeasibility suggest a feasible and approximately optimal solution, but the absolute percent error on the objective is not close to 0%, we may guess one of three things. Either the solution to the IOC problem is non-unique and there exists an objective function that describes the traffic controller better or the optimal control problem is at a local optimum or both of these are simultaneously true. In other words, solving the optimal control problem many more times may reveal the globally approximately optimal solution. However, since the KKT conditions are only necessary conditions, the existence of a recovered objective function that can estimate the true system's strategy better is also possible.

Again, it is important to note that a locally approximately optimal solution might be all that is needed to approximately mimic the traffic controller’s policy. Therefore, using a non-convex feature set and solver is useful because we can define various features that make the range of objectives that can be recovered much larger. However, using a convex feature set is beneficial since we are always guaranteed to obtain the best possible solution for a given problem.

Now, we solve the same IOC problem using a convex feature set (all features from Table 3.1) in order to test the performance of the method when the feature set does not explicitly contain the non-convex features used in generating the observed trajectory. Since the optimal control problem is now convex and, as a result, the KKT conditions are necessary and sufficient, we only need to solve it once. The results (Figures 3.7b and 3.7b) show that the convex feature set actually does a little bit better than the non-convex feature set that was previously considered in mimicking the switching time policy of the observed traffic controller ($e_\delta = 6.30\%$). The absolute percent error on the objective ($e_{\text{obj}} = 0.85\%$) is also smaller. The infeasibility error ($e_i = 8.00 \times 10^{-18}\%$) and optimality error ($e_o = 0.0006$) are also small and suggest a feasible and globally approximately optimal solution.
Figure 3.7: Observed and simulated phase lengths for a delay-minimizing signal.
Figure 3.8: Observed and simulated switching times of a delay-minimizing signal.
3.6 Summary

Traffic controllers are an integral part of any modern urban traffic system and understanding their control strategies is essential to predicting future states of traffic from observed data. However, it is generally challenging to procure the control strategies of traffic controllers in large scales when it is available, and in other cases it is impossible, due to proprietary algorithms or human–based traffic controllers.

The work in this chapter extends the numerical validation provided in [10] by providing an experimental validation of the technique. We showed that the inverse optimal control problem for a traffic signal controller can be solved via convex programming to recover the control objective of an observed traffic controller. In particular, we showed that for a pre–timed signal and a delay–minimizing signal, we were able to recover a control objective that produces a simulated trajectory that is similar to the observed trajectory. We also introduced performance metrics that can be used to evaluate the quality of solutions to the IOC problem. The next steps to be taken include testing the method on other types of traffic controllers, including human controllers. The eventual goal is to be able to create a real–time traffic controller strategy estimator for the TrafficTurk system. The possibility of coordination amongst traffic signals might also be explored.
Large-scale field deployments of Traffic Turk

Traffic Turk has been deployed at more than 100 intersections around Urbana-Champaign, IL to monitor traffic at the 2012 Homecoming football game, in New York City following Hurricane Sandy, and at the Farm Progress Show 2013 in Decatur, IL.

A contribution of this thesis is the design of these three large-scale deployments. For the purpose of providing insight into the experimental design, implementation, and data analysis associated with deployments of Traffic Turk, this chapter describes a deployment at the Farm Progress Show, 2013 in detail and provides data analysis as it relates to the objectives of the deployment.

4.1 Overview

The Farm Progress Show is the national’s largest outdoor agricultural equipment exhibition and is held biennially in Decatur, IL. The event attracts anywhere between 125,000 – 175,000 people over the 3 days of the event, and creates significant congestion in the area. In 2013, the event was held during August 27-29 and Traffic Turk was used to collect traffic data on August 27 and 28, 2013. The data collected by the application will be used to assist in the analysis of the current traffic management plan for the show. This is critical as any improvements would reduce the delay experienced by visitors to the show as well as reduce queues on the interstate highways and as a result
reduce opportunities for high-speed rear-end collisions. Three major research tasks were associated with this particular deployment. They were:

1. Data collection during the Farm Progress Show 2013.
2. Data analysis for improvements to future traffic management plans.
3. Evaluation of the technology and its potential for greater use.

In this chapter, we detail each research task, its execution, results and potential improvements for future deployments. Finally, we attempt to provide future direction and scope for TrafficTurk.

4.2 Data Collection at the Farm Progress Show

4.2.1 Motivation for the deployment

Traffic generated by the Farm Progress Show creates significant congestion between the hours of 7am and 10am on August 27 and August 28, the two peak days of the show. Using TrafficTurk, the traffic levels during the peak hours at 31 critical locations in the surrounding area on both days will be measured. The locations of the traffic measurements are shown in Figure 4.1. Data will be collected at all 31 intersections simultaneously, to provide a detailed picture of the network wide traffic conditions on the main routes into the show. The locations for data collection were selected in collaboration with the Macon County Highway Department.

4.2.2 Detailed summary of the deployment

4.2.2.1 Testing the system

The collection of data using TrafficTurk requires an underlying map over which counting locations are overlaid. This is necessary in order to uniquely identify each counting location as a location on the physical road network. Moreover, each counting location has specific road geometry and the collection of turning movement counts requires knowledge of these geometries. For example, collecting turning movement counts at a 3-way intersection (T-intersection) is
Figure 4.1: Data collection locations and typically congested routes during the Farm Progress Show 2013.
different from collecting turning movement counts on a highway. Therefore, in order to ensure that during data collection, the application is able to correctly display the road geometry and location on the road network, we thoroughly tested the application at each counting location.

In addition to this, we collected critical information such as feasible and safe locations for an individual to collect data from, the quality of the cell-phone network at each location, any errors in the application related to display of the street-names or road geometry, etc.

After this test, the map used in the application (OpenStreetMap) was edited to correct erroneous road geometry, incorrect street-names were changed to match the street signs on the physical road network and any relevant modifications to the application were made.

This ensured that collecting data at every intersection would be possible, would be reliable and that the user-experience would be smooth and intuitive. The intersections where cell-phone network quality was seen to be low were also flagged for use in the event management, which will be described in a following section.

Further, the entire system was tested for accuracy and completeness of data. In other words, we developed and tested the system to ensure that every single vehicle that was recorded with the application was accounted for in the database and that each data point was stored accurately.

This involved creating different data collection scenarios, entering a known sequence of movements into the application and checking whether it was reflected exactly in our database. Different scenarios include combinations of various data connection qualities and various counting location types.

An example scenario is as follows: “A user starts collecting data at the intersection of two two-way streets but loses the data connection shortly after, but continues to collect data. The data connection is re-gained later on.”

4.2.2.2 Recruitment of data collection associates

In order to collect data simultaneously at all the desired counting locations, a sufficient number of people needed to be recruited to collect traffic data during the event. We will refer to the recruits as data collection associates. Each data collection associate would be compensated for their time and effort contingent
on the fact that they completed their assigned tasks.

One of the main objectives of the deployment is to collect a complete data set for each counting location. In order to ensure that this is possible, the recruitment of data collection associates needed to consider the fact that several people might not complete their assigned tasks due to a number of reasons (not able to travel to counting location, phone not working, illness, change of plans, not prepared for the weather, forgetting about the tasks, not waking up on time, etc.). Thus, we planned to recruit many more data collection associates than counting locations.

The process of being recruited as a data collection associate for the Farm Progress Show started at TrafficTurk’s website (www.trafficturk.com). The website was modified just before the recruitment period started in order to quickly direct people to the registration page. The website’s structure was such that one would first enter a page with detailed information about the responsibilities of being a data collection associate. Then, if a person decided to sign-up for the opportunity, they were directed to a form that they must submit. The responses to the form were directed to a spreadsheet in Google Drive from which we were able to filter the potential recruits.

Certain questions in the form were designed to obtain information that is critical in determining whether a person will be able to successfully carry out their duties as a data collection associate. For example, one questions asks about the phone that they own. Since TrafficTurk is only available for smartphones that run the Android operating system, any response that indicated otherwise disqualified that person from being a data collection associate. Note that despite providing information about the requirement that their phone must run on Android, we received several responses that indicated otherwise. Similar questions were asked to roughly gauge the ability of the person to collect traffic data.

The strategy to direct web traffic to the TrafficTurk website was basically to raise awareness about the opportunity on local media including television and newspaper. A number of local groups were also contacted to promote the opportunity within the group. In total, we received 75 responses through the website. Out of these, emails were sent to 70 individuals asking for a confirmation of their desire to participate in the deployment. We received confirmations
from 51 people. Eight people canceled their participation between signing-up and attending a training session (detailed in Section 4.2.2.3). Two people did not show up to their assigned training session. Overall, we had 41 people recruited and trained as data collection associates.

4.2.2.3 Training of data collection associates

Each data collection associate was required to attend a training session in order to participate. There was a choice of four training sessions that one could attend. Everyone indicated the training sessions that they could attend on the sign-up form and was assigned to a training session.

Overall, 41 people attended a training session. Out of these, 31 people were assigned to counting locations and 10 were assigned to be backup data collection associates. This meant that they would report to the deployment control center where in case of need, they would be sent to a counting location. The need may arise because a data collection associate’s phone’s battery is exhausted, or a data collection associate fails to show up on time, etc. The backup data collection associates would be compensated if they were available to act as a backup for the entire duration of data collection.

Training sessions covered essential topics like personal safety, how to use the application to collect traffic data, the importance of being on time, how to get paid, etc. The training session was structured in form of a presentation followed by some mock data collection.

However, as people arrived at the training session, they were encouraged to install the TrafficTurk application on their phone if they had not already done so. They were also given help if they needed it by training session staff. They were also required to sign-in to the training session with training session staff who would ask them to go through the statement of informed consent and talent release form, and sign them before proceeding with the rest of the training session.

The trainees also practiced collecting data with TrafficTurk by watching a video of a counting location. There were multiple videos available for different types of counting locations.

Just before the training session, the Macon County Highway Department marked each counting location on the physical road network with a pink spray.
This indicated where one could park their vehicle safely and collect data with TrafficTurk. This information was given to the trainees in the form of a picture of the counting and parking location. Each trainee was also given a bright T-shirt and a placard that identified them as part of a research study to the general public and authorities. In addition to this, each trainee was also given an information packet that consisted of three main documents.

The first was a the-day-of handout which had information about the deployment, the times, locations, phone support phone number, tips to make the experience more comfortable, troubleshooting for common issues, etc.

The second document was a list of safety considerations that simply reiterated the safety instructions presented in the presentation. Finally, a map of the 31 counting locations with each data collection associate’s specific location highlighted was also included.

Backup data collection associates were given all of the above except a picture of their counting location and a highlighted map. Instead, they were given an un-highlighted map of all 31 intersections that they could be asked to go to during the deployment.

4.2.2.4 Event managers

The role of event managers during the data collection period was to ensure that data was being collected at all 31 intersections and to provide the data collection associates with the assistance needed to complete their assigned tasks. Since there are a number of different tasks to be completed in managing the event, event managers performed three distinct roles – data monitor, phone operator and field support.

Data monitor  The role of the data monitor is to ensure that all counting locations have active data collection associates collecting data. In order to do this, data monitors use a tool known as the monitoring tool. This tool runs in an internet browser and is able to display the current status of each counting location.

The tool displays various attributes of the data being collected at each counting location as well as attributes of the user and his/her phone. This is helpful because it is a quick way of identifying current issues and predicting
future ones. For example, one could easily identify the case where data is
not being collected at a counting location and one could estimate whether a
phone will run out of battery life before the end of the counting period. The
data monitor used this information and made decisions regarding issues. For
example, the could decide that someone needs to be checked on in the field,
or that someone needs to be called, or that a backup data collection associate
needs to be sent to replace someone.

For this particular deployment, our team consisted of two data monitors.
The data monitors also constantly updated a Google spreadsheet that was
accessible in real-time by everyone at the event control center. This helped
everyone stay abreast of the ongoing issues and allowed others to jump in to
help if needed.

**Phone operator**  Phone operators complemented data monitors by helping
to resolve issues over the phone. They were equipped with a Google Voice
account from which they could call or text any data collection associate. They
also received messages from data collection associates and field support. These
messages were used to keep track of data collection associates’ issues and the
location and status of field support. They were also responsible for delivering
phone support to data collection associates and communicating with field
support. For example, a phone operator could call a data collection associate
who has a problem with the application and guide them through how to fix
the issue.

If the phone operators were not able to resolve the issue over the phone,
they asked field support to help the data collection associate in person. For
this deployment, two phone operators were used.

Each phone operator used a different Google Voice account (and hence,
phone number). Half of the data collection associates were given one Google
Voice phone number on their *day-of-handout*, and the other half were given
another. This way, each phone operator (and each data monitor) only had a
subset of the counting locations to monitor and provide support to.

**Field support**  Due to the size of the area covered by the 31 counting
locations, all field support personnel were equipped with a car and backup
phones. Backup phones were available in case a data collection associate’s phone stops working due to an exhausted battery or any other issue.

Field support personnel were instructed to send a text message to the phone operators each time they moved their location in order to help the data monitors and phone operators make decisions on field support. In this deployment, there were three field support personnel, each with their own area of operation. This division of the covered area was necessary so that each car could quickly get to any counting location in their designated area.

To facilitate this further, all field support personnel were given maps of their area with the counting locations and congested routes highlighted. They were instructed to avoid congestion in order to quickly provide field support. This required that they were very well acquainted with their area of operation and were encouraged to explore the area before the data collection period.

**Event control center**  The event control center housed the data monitors, phone operators and backup data collection associates. This was the hub for communication and decision making. It also served as the place where data collection associates could come to pick up their compensation.

For this deployment, the event control center was the Macon County Highway Department office in Decatur, IL. This location was sufficiently far away from the Farm Progress Show so that there was no congestion in and around it, yet sufficiently central such that one could go to most counting locations quite quickly.

The most important pieces of equipment at the event control center were computers that were used to monitor the incoming data and to make phone calls through Google Voice. A fast, reliable internet connection was a supplementary requirement.

Another important feature of the event control center was the real-time sharing of event management tools. Google Drive/Google Docs allows multiple users to access and edit documents while they are being edited in real-time and therefore, it is a great way to keep the entire event control center updated with the latest information on the deployment.

An issue tracker tool to keep track of data collection associates’ issues, and a field support tracker to keep track of the locations and availability of field
support personnel were used by all event managers at the event control center. The monitoring tool was also used by everyone at the event control center since multiple instances of it could be opened in an internet browser.

The event control center was also the location where compensation was paid to the data collection associates at the end of the second day of data collection. Before paying each data collection associate, event managers made sure that the database contained all the data from each data collection associate for both days of data collection. There was a feature in the monitoring tool that showed whether we had a complete data set from each data collection associate.

In case someone’s data had not made it to the database due to any reason (no internet connection at their counting location, something wrong with the phone, etc.), the event managers restarted the application and connected the phone to the WI-FI internet connection at the event control center. This ensured that any collected data that was locally stored on the phone but not yet sent to the TrafficTurk database was sent. Once all the data was seen in the database, the data collection associate was paid and thanked for their time and effort.

4.2.2.5 Data collection Day 1 – August 27, 2013

The first day of data collection started early for the TrafficTurk team. Being based in Champaign, IL, the event management team departed to the event control center in Decatur, IL at 4:45am. One team member stayed back to keep an eye on TrafficTurk’s back-end systems (Servers, database, monitoring tool, map etc.).

Approximately 50 minutes later, the team reached the event control center and started setting up for the day’s data collection. This included setting up computers, screens, a waiting area for backup data collection associates, etc. The systems and tools at the event control center were also tested thoroughly.

In the meantime, field support personnel directly went to their areas of operation and explored their area. This exploration was required in order for them to be familiar with their area before the start of data collection. It was also a chance for them to find a central location in their area where they could wait to receive field support tasks from the event control center. One of the field support personnel described this exercise as, “absolutely necessary since
some roads were closed and others had street names that were different from the maps”.

At 6:45am, 15 minutes before the start of data collection, the phone operators sent a mass text message to all data collection associates. This text read as follows:

“Good morning! The TrafficTurk test will begin in 15 minutes. Please go directly to your counting location, and have a great morning!”

This text was intended to remind anyone who had either forgotten about the day of data collection or had not woken up on time about the day’s tasks. This also gave data collection associates with last minute change of plans a chance to reply and let us know that they cannot make it, so that we could arrange for a backup data collection associate to take their place.

Then, at 6:55am, another text was sent to everyone who had not been seen on the monitoring tool at their counting location. This text read:

“We do not see you at your counting location yet. Please let us know if you are on your way.”

Then, at 7:00am, a final text was sent to anyone who had not started collecting data or had not replied to earlier texts confirming that they are on their way.

“We do not see you at your counting location, please reply: I am on my way, or I am already here. Thank you.”

After this text, most people had either shown up at their counting location or had confirmed that they were on their way. A few people had not replied nor shown up at their counting location.

At this time, field support personnel were sent to these locations to confirm whether they had indeed not shown up or whether their phone had no network connectivity and as a result did not receive texts or send any data to the TrafficTurk servers.

It turned out that on this day, two data collection associates did not show up and they were promptly replaced by backup data collection associates.

After this busy initial period, field support personnel made their way to each counting location and ensured that every data collection associate was wearing their bright T-shirt, had a placard displayed and that they were not experiencing any problems. Any issues that came up were resolved by the field
support personnel.

Soon after, all counting locations were covered and the data was being monitored continuously. There were relatively a very small number of issues on Day 1 as compared to previous deployments.

At the end of the data collection period, another mass text message was sent to all data collection associates in order to inform them that the day’s work was done, to thank them for their effort, and to ensure that they have safely departed their counting location.

“Thank you very much for your participation. You have done a great job in helping us out. Please send us a text to confirm that everything went well and you have left the counting location.”

In total, 87778 vehicles were recorded between 7:00am and 10:00am on Day 1. One data collection associate collected data at the wrong counting location. They collected data at the intersection of IL 48 and Boyd Road instead of at the I-72 eastbound (EB) ramp onto IL 48. This meant that there was no data collected at I-72 EB ramp onto IL 48 until this mistake was realized. In the end, only 27 minutes of data collection occurred at this counting location.

The highest number of vehicles recorded at any counting location was 8105 vehicles on I-72 at Greenswitch Road and the smallest number of vehicles recorded at any counting location was 575 at the intersection of Brush College Road and College Park Road. A total of 5447 minutes of active data collection occurred on Day 1, which is 94.6% of the total counting time that could have been covered. The counting location with the lowest flow rate was the intersection of Brush College Road and College Park Road (192.6 veh/hr) and the counting location with the highest flow rate was I-72 at Greenswitch (2702.3 veh/hr). The average flow over the whole network was 981.7 veh/hr.

4.2.2.6 Day 2 – August 28, 2013

Day 2 of data collection started off in a similar fashion except that the equipment at the event control center was already setup. The field support personnel were also already familiar with their area and the data collection associates were familiar with their counting location and their duties. This ensured that the start of the day’s data collection was smooth.

The same text messages were sent as Day 1 and very quickly, all counting
locations had a data collection associate collecting data. After the end of Day 1’s data collection period, a quick debrief helped the event management team look for any issues and correct them on early on Day 2.

One of the field support personnel coordinated with the police department to move a data collection associate to another counting location. This was done because the counting location was not ideal to record data from. As congestion built up, the queue blocked the data collection associate’s view of part of the intersection and therefore, they had to be moved to another location.

Three data collection associates reported that they were indeed at their counting location collecting data; however, due to poor network quality at their locations, the monitoring tool did not show any of their collected data. Field support personnel were sent to confirm their locations and a note was made to connect their phones to an internet connection at the end of the day’s data collection.

Apart from that, the Macon County Highway Department suggested that turning movement counts at the interchange between I-72 WB and US 51 were important. Therefore, shortly after the day’s counting started, a backup data collection associate was sent to that location and started collecting data.

On Day 2, a total of 90384 vehicles were recorded. Again, the counting location with the least recorded vehicles and smallest flow rate was the intersection of Brush College Road and College Park Road (660 vehicles at 221.8 veh/hr). Interestingly, the counting location with the highest number of vehicles recorded was the intersection of Mound Road and US 51 (6850 vehicles). This intersection also had the highest flow rate out of all the counting locations (2289.1 veh/hr).

Data was actively collected by the data collection associates for 5483 minutes, which is 98.3% of the total possible data collection time between 7:00am and 10:00am. The average network flow rate was 954.3 veh/hr.

4.2.3 Deployment statistics

Here are some statistics that illustrate the scale of the deployment:

- 96.5% of the total data collection time over all counting locations was
• A total of 178162 vehicles were recorded over more than 182 man-hours of data collection.

• 10 issues were resolved over the phone.

• 5 backup data collection associates were sent out to collect data.

• 2 issues were resolved by field support personnel.

• 0 backup phones were used.

4.3 Potential improvements for future deployments

Some potential improvements for future deployments are:

• The number of counting locations can be increased significantly to get a more complete picture of the traffic movement. This would definitely involve a more aggressive and prolonged recruitment process since many more data collection associates would be required. Needless to say, it would also cost more money. However, having more counting locations would provide some redundancy in the collected data and this could be used to estimate the state of traffic with more certainty and also help in the evaluation of the technology (compare differences in collected data between two or more data collection associates etc.).

• An automated system to send mass text messages could be incorporated into the monitoring tool. This way, event managers do not have to individually check each counting location and send text messages appropriately. This would enable much larger deployments.

• A GPS enabled tracking system for field support personnel would eliminate the need to continually use the field support management tool to keep a track of the location of field support personnel. This would reduce the work load of the event management staff and as a result facilitate decision making and better monitoring of incoming data.
- An online payment system for data collection associates will eliminate the need for the organization and execution of an in-person compensation session. This will reduce the work load on all participants and also facilitate much larger deployments of the technology.

### 4.4 Traffic Data Analysis

Due to the inherent nature of how TrafficTurk data is collected, it is susceptible to the introduction of biases due to human error. The user interface of the application is designed to minimize this error; however, it does not completely eliminate it. Therefore, some sort of normalization of the data is required.

This section describes a method that takes advantage of the concept of mass conservation to build an optimization problem that ensures that the data-set is normalized and feasible. For example, if one of two data collection associates on either end of a road segment misses vehicles in their data collection, it is possible that over a period of time, we may see a negative queue length develop on the road connecting the two counting locations. Of course, this is not feasible. The same road might have a queue that far exceeds its capacity in the opposite direction of traffic flow. In order to ensure that these situations do not arise when we analyze the data, the normalization, in the form of a solution to an optimization problem is performed. The methodology is described in Section 4.4.1.

The normalization was carried out for all links (road segments) in the network that had a data collection associate at each end. However, some links had very high rates of vehicles exiting the link in between the two data collection location and therefore, the mass conservation did not hold for these links.

For example, the segment of Brush College Road between Mound Road and College Park Road seemed to have an entrance into the parking area for the show, and this created a huge outflow of vehicles from the link. The results of the normalization include correction factors for each of the sensors’ data and also the evolution of the storage of vehicles on a segment of the road over time.
4.4.1 Normalization of data using mass conservation of vehicles

If sensors (data collection associates) $i$ and $j$ have some cumulative error associated with their data, suppose that we can multiply their cumulative vehicle counts by some factors $\delta_i$ and $\delta_j$ to correct their data to the true counts. Then, for each link (segment of road) $l$ that lies between sensor locations $i$ and $j$, the following equation must hold:

$$s_l(k + 1) = s_l(k) + \delta_i(q_{i,\text{in}}^l(k)) - \delta_j(q_{j,\text{out}}^l(k)) \quad \forall i, j, k \text{ and } l,$$

where $s_l(k)$ is the storage of vehicles on link $m$ at the beginning of time-step $k$ and $q_{i,\text{in}}^l$ and $q_{j,\text{out}}^l$ are the inflow and outflow onto and out of link $l$ during time-step $k$ at counting locations $i$ and $j$ respectively. Moreover, we know that the storage of vehicles on a link cannot be negative and it cannot exceed the capacity of the link (jam density). Therefore, we introduce the constraints,

$$0 \leq s_l(k) \leq c_l \quad \forall l,$$

where $c_l$ is the capacity of link $l$, which can be calculated based on the physical properties of the road segment including length, number of lanes, jam density, etc. Additionally, we impose the constraint that $\delta_i$ cannot be negative.

$$0 \leq \delta_i \quad \forall i.$$

Now, suppose we assume that most of the sensors are being fairly accurate, we would expect most of the values of $\delta_i$ to be close to one. In addition, we can also reasonably assume that the storage on all links at the end of the data collection time period is low.

Keeping this in mind, we can build the objective function and write the optimization problem for a network with $n$ sensors and $m$ links as:
\[ \min_{\delta,s} \sum_{i=1}^{n} (1 - \delta_i)^2 + \gamma \sum_{l=1}^{m} [s^*_l(k_f) - s_l(k_f)]^2 \]

subject to:
\[ s_l(k + 1) = s_l(k) + \delta_i(q_{in}^l(k)) - \delta_j(q_{out}^l(k)) \quad \forall i, j, k \text{ and } l \]
\[ 0 \leq s_l(k) \leq c_l \quad \forall i, j, k \text{ and } l \]
\[ 0 \leq \delta_i \quad \forall i, \]

where \( s^*_l(k_f) \) is the desired storage on link \( l \) at the beginning of the final time-step \( k_f \) and \( \gamma \) is a chosen weighting factor.

In general, the results of the normalization showed that apart from three sensors, all sensors had \( \delta \) values between 0.95 and 1.05. We presume that the three sensors with \( \delta \) values outside this range were at the end of links that had vehicles leakages in the middle of the link due to the existence of a parking lot, for example.

The other result of the normalization is the evolution of vehicle densities on each of the links in the measured network as a function of time. These results are provided in the analysis of traffic in Sections 4.4.2 and 4.4.3 in the form of map graphics.

### 4.4.2 Traffic from the north and west

Figure 4.2 shows the different routes to the show from the west and north. The main entrance routes to the show from the west include:

- Follow I-72 EB till the interchange on Route 48 and head south on Brush College Road.

- Exit I-72 EB at Rt. 121 and follow 121 till Mound Road. Head east on Mound Road to Brush College Road.

- Exit I-72 EB at the interchange at US 51 and head to Mound Road. Head east on Mound Road to Brush College Road.

- Follow US 51 to I-72 EB. Exit at Route 48 and head south on Brush College Road.
Follow US 51 to Mound Road and head east on Mount Road to Brush College Road.

The most important route to consider in this analysis is the series of roads between Route 48 at I-72 EB Ramp all the way to the intersection of Brush College Road and Mound Road. This flow on this road segment determines whether there will be a queue on the I-72 EB exit ramp at Route 48. A queue on this ramp gives rise to a dangerous situation where a high speed road has a long queue of slow moving vehicles. This increases the chances of a high speed rear-end collision and should be avoided if at all possible. Therefore, we analyze the storage of the road segments along this route in 10 minute time periods between 7:00am and 10:00am.

For the route between I-72 EB ramp on Route 48 and the intersection of Brush College Road and Mound Road, the estimate average state of traffic in between 7:10am and 7:20am is shown in Figure 4.3.

The road segments marked in green have an estimated average density of vehicles that is less than one-fifth of the jam-density of the road segment. Road segments marked in yellow have average density values between one-fifth...
Figure 4.3: A visualization of densities on Route 48 and Brush College Road on August 27 (left) and August 28 (right), 2013 between 7:10am and 7:20am

Figure 4.4: A visualization of densities on Route 48 and Brush College road on August 27 (left) and August 28 (right), 2013 between 7:40am and 7:50am

and two-fifths of the jam-density. Orange indicates that the average density of the road segment is between two-fifths and three-fifths of jam density and red indicates that the density is above three-fifths of jam density. The figure shows that there is already a very high density of vehicles on Route 48 on both days, which indicates that most likely there is a queue building on the I-72 exit ramp. Similarly, the road segment just before the entrance to the show also has a high storage value.

It can also be noticed on August 28, 2013 (we will refer to this day as Day 2 and August 27, 2013 as Day 1 for the remainder of the thesis) that there is low density on the road segment between the WB and EB ramps which indicates that there is relatively less traffic coming in from the east onto I-72 WB ramp than from the west onto I-72 EB ramp.
In the meantime, there is also low storage on Mound Road between US 51 and Greenswitch Road on Day 2 whereas on Day 1, these road segments have a higher density. It is clearly seen that since the intersection of Mound Road and Brush College Road has to serve two streams of incoming vehicles, there is high storage just upstream of the intersection on both Brush College Road and Mound Road on both days.

Then, in Figure 4.4, we see the densities on Route 48 and Brush College Road during the time period between 7:40 and 7:50am. A reduction in densities is apparent on Route 48 on Day 2 between Brush College Road and the I-72 WB ramp. The density on Brush College Road has decreased on Day 1, but on Day 2, there is a similar density.

Interestingly, the density on Mound Road between Brush College Road and Greenswitch Road has decreased on Day 2, but has increased on Day 1. From 8:00am onwards, the densities on all these road segments reduce for the remainder of the data collection time period. The average densities between 8:10 and 8:20am are shown in Figure 4.5.

The western approach from Mound Road does not get highly congested for the duration of the data collection. Specifically, the road segment between US 51 and Greenswitch road maintains a density that is less than one-fifth of the jam density throughout the three hours of data collection. This suggests that there is available capacity that can be filled with vehicles.

If this available capacity is filled, i.e. a portion of east bound vehicles on I-72 are re-routed to enter Brush College Road from Mound Road, it may
reduce the queue on the I-72 EB ramp at Route 48. However, there seem to be bottlenecks at the intersections of Mound Road and Route 48, and Mound Road and Brush College. These may induce spill-back onto the intersections of Mound Road with Route 48 and Greenswitch Road respectively.

The negative effects of this spill-back can be reduced if cross traffic on Route 48 and Greenswitch Road is temporarily stopped. This re-route can be made by encouraging vehicles to exit I-72 EB at the interchange of Route 121. Then, vehicles can make their way onto Mound Road and eventually to the show.

However, the fraction of vehicles that are going to show and can be taken off at the Route 121 interchange needs to be known to estimate whether this particular re-route will indeed reduce the queues at Route 48. Moreover, the re-route also should ensure that a queue on I-72 EB does not instead form at the interchange of Route 121. In order to do this, we analyze a similar visualization of densities on the road segments around the 121 interchange.

It can be seen from Figure 4.6 and Figure 4.7 that the densities on the road segments just downstream of the I-72 exit ramps at the 121 interchange stay fairly low at the beginning of the day’s data collection. From 7:30am onwards, the densities on these road segments does not increase. By 7:40am on Day 1, the road segment of Route 121 between I-72 WB ramp and Mound Road has decreased to a level that is less than one-fifth of the jam density and this does not change for the remainder of the day’s data collection period.

On Day 2 however, this road segment stays at average density values between one-fifth and two fifths of jam density for the majority of the data collection.
Figure 4.7: Evolution of densities on August 28, 2013 on Route 121 around the interchange with I-72

collection time period. This suggests that there is still some vehicular storage space available on these road segments.

Now, in order to check how much traffic can potentially be pulled off I-72 EB at this interchange, we analyze the total turning movement counts and the arrival rates at the counting locations located on the I-72 WB and I-72 EB ramps and the intersection of Route 121 and Mound Road. The calculation that estimates the amount of traffic that can be taken off at this interchange is the sum of the number of vehicles on I-72 EB at the US 51 overpass and the number of vehicles exiting I-72 EB at 121 minus the number of vehicles entering I72 EB at 121. This yields a total of 3515 vehicles on I-72 EB at 121 on Day 1 and 3335 on Day 2.

Out of these vehicles, we must estimate the fraction that is going to the show. In order to do so, we subtract the sum of vehicles entering and exiting I-72 EB at US 51 from the number of vehicles exiting I-72 EB at Route 48.

The assumption made here is that most vehicles that exited I-72 EB at Route 48 would have passed through the Route 121 interchange. We also assume that if a vehicle took an exit onto US 51, they are most likely not heading to the show and that most vehicles taking I-72 EB from US 51 are going to the Route 48 exit.

While this assumption may not hold exactly, it helps us estimate roughly how many vehicles can be re-routed at Route 121. Note that the quantity (number of vehicles taking exit from I-72 EB at US 51) was not available due to human error in the data. One data collection associate collected the wrong stream of traffic. However, we estimate this quantity to be small because the number
of vehicles traveling southbound on US 51 at the intersections of Forsyth and Mound Road are very similar.

A similar error on Day 1 at I-72 EB ramp at Route 48 keeps us from calculating this quantity for Day 1, therefore we provide this calculation only for Day 2. The number of vehicles on I-72 EB at 121 that are going to the show on Day 2 was calculated to be 1941.

An assumption that can be made about the 1941 vehicles that are going to the show is that their arrival rates follow the same distribution as the traffic on I-72 EB at US51 (the closest counting location on I-72). Figure 4.8 shows the estimated arrivals of show-going vehicles if a complete re-route is instantiated.

Now, we check whether this re-route will create a queue on I-72 EB at the 121 interchange. In order to do this, we calculate the peak flow with the re-route in effect just downstream of the I-72 EB exit and check to see if this number breaches the saturation flow rate of the roads involved in the re-route. The peak flow rate of eastbound vehicles on Route 121 just downstream of the I-72 EB is around 800 vehicles/hour.

Now, we may check whether the ramp has a high enough saturation flow rate to support the inflow of vehicles. Since the ramp exits onto Route 121
and show-bound vehicles must take a left turn from the ramp onto Route 121, the calculation of the saturation flow rate of this movement must be done with a left-turn adjustment factor.

The method followed to calculate the capacity is taken from [30]. Equation 10.5 in [30] is used to calculate the saturation flow rate. This equation holds for signalized intersections and it is used because a major re-route would require some type of control (possibly hand directed) in order to ensure that show-bound vehicles can safely take left turns onto Route 121 from the I-72 EB ramp. The saturation flow rate is calculated under the condition of a shared left-turn lane with protected left turns. The estimated saturation flow rate of the ramp is calculated to be 1277 vehicles/hour.

Other calculations of capacity are also required here since we must ensure that the entire route has a high enough saturation flow rate to support the inflow of vehicles from both I-72 EB and Route 121 EB. The capacity of the intersection at Route 121 and Mound Road is also calculated using equation 10.5 from [30].

We calculate this quantity with the assumption of a one lane road because all show-bound vehicles must take a left at Mound Road, and therefore it would be unreasonable to assume that more than one lane can be used for storing vehicles.

Again, we assume that the intersection is signalized because of the large number of left-turns that need to be made onto Mound Road from Route 121. The saturation flow of left turns at this intersection is estimated to be 1805 vehicles/hour.

The calculations indicate that the saturation flow rate on Route 121 is not breached and therefore, this re-route could be feasible without a large queue building up on the ramp of the interchange, providing that there is some form of traffic control implemented at these two major turns in the route that does not reduce the estimated saturation capacities. Another quantity that is important to check is the storage capacity of the route.

The calculation of this quantity suggests that there is enough capacity on the road to store 98% of show-going vehicles on the route at the same time (1901 vehicles). However, this re-route could be only partially implemented in order to use the storage capacity of Mound Road as well as the storage capacity
on Route 48 and Brush College Road. An effective way of splitting show-going traffic will be required in order to make this partial re-route effective.

Another re-routing option that exists is to partially re-route traffic on I-72 EB to take exits at Route 121, US 51 and Route 48. However, this option may not be the best since there is already a high flow of vehicles on US 51 in the south-bound direction (Figure 4.9) and any additional merging traffic might induce queues on both US 51 and the I-72 EB exit ramp onto US 51.

### 4.4.3 Traffic from the east

The main routes into the show from the east are shown in Figure 4.10. They include:

- Follow I-72 WB and exit at Argenta Road. Then, join Oakley Road and head south till Cerro Gordo Blacktop. Then head west and eventually make your way to Reas Bridge Road.

- Follow I-72 WB and exit at Route 48. Turn onto Brush College Road and head south.

- Follow Oakley Road northwards towards Angle Crossing Road. Head west onto Reas Bridge Road.
Most of the traffic (approximately 70%) from the eastern and southern approaches arrives from the east on I-72 WB. The recommended route for vehicles traveling to the show on this road is to take the exit at Argenta Road and follow signs through Oakley Road onto Reas Bridge Road. Additionally, there is another stream of show-bound vehicles that enter Oakley Road from the south.

As it can be seen from Figures 4.11 and 4.12, the most congested 30 minutes on these approaches are the first 30 minutes of data collection on both days. After these 30 minutes, this route becomes progressively less congested the majority of the roads have average densities that are less than one-fifth of the jam density. This suggests that there is more capacity available on this route that could be utilized. The utilization of this storage capacity depends on the amount of traffic that follows the recommended route to the show from the east.

The data suggests that only 65% of show-bound vehicles take the recommended route on both days of data collection. The rest of the show-bound traffic takes the exit at the Route 48 interchange.

This is not ideal since this congests the already highly demanded Brush College and Route 48 roadways and potentially increases the queuing on the exit ramps from I-72 onto Route 48.
Figure 4.11: The most congested 30 minutes on the eastern and southern approaches to the show on August 27, 2013

Figure 4.12: The most congested 30 minutes on the eastern and southern approaches to the show on August 28, 2013
Figure 4.13: The estimated arrivals onto Argenta Road from I-72 WB if all show-bound vehicles followed the recommended route

In order to check whether it is possible to have all show-bound traffic take the recommended route (i.e. will or will it not induce a queue on the Argenta exit ramp?), we can estimate the demand and the storage capacity along the route. The demand at the Argenta exit in the case where all traffic from the east takes the recommended route is shown in Figure 4.13 in terms of flow rates.

This figure is only for Day 2 since the data for the intersection of Route 48 and I-72 WB ramp is not available for Day 1. The data collected at the intersection of Argenta Road and Illiniwick Road (4.14) is taken as a proxy for the show-bound traffic taking the Argenta exit since a data was not collected on Argenta Road at the I-72 WB exit ramp. We assume that the number of vehicles that joined Argenta road from the north of the I-72 WB exit ramp remains the same as on the measured days. We also assume that the distribution of arrival rates at the Argenta Road remains the same.

The number of show-bound vehicles over the entire data collection period is estimated to be 1857. This turns out to be 129% of the storage capacity in
the route between I-72 WB exit ramp on Argenta to the intersection of Stare Road and Reas Bridge Road.

Therefore, at one time, all vehicles taking the Argenta Road exit will not fit between I-72 WB ramp on Argenta Road and the intersection of Reas Bridge Road and Stare Road. However, the time required to fill the entire route to jam density is approximately 2 hours, which is a long enough time interval for a large majority of vehicles to traverse the 8.2 mile route.

On the days of data collection, the densities on the road segment just upstream of the Argenta exit on I-72 WB are shown in Figures 4.15 and 4.16. On both days, the highest densities are seen at the beginning of the data collection period. On Day 1, the average density is between one-fifth and two-fifths of the jam density between 7:15am and 8:45am. Then, the density is less than one-fifth of the jam density for the remainder of the data collection period. On Day 2, we see a higher density at the beginning of data collection. For the first half hour of data collection between 7:00am and 7:30am, the density on the road segment just upstream of Argenta exit on I-72 WB is between two-fifths and three-fifths of the jam density.

Then, for the next 50 minutes, the density is between one-fifth and two-fifths of jam density. After 8:20am, the density goes to less than one-fifth of jam density. These figures suggest that there is a moderate amount of congestion at the approach to this exit on Day 2 even though approximately
65% of show-bound vehicles took this exit.

Therefore, suppose all vehicles headed to the show took the recommended route, we could expect more congestion in this road segment. In order to check the magnitude of flow can be handled by this exit ramp without causing long queues on the exit ramp, we check the potential capacity.

We assume that the majority of traffic on the ramp is show-bound and is turning left from the exit ramp onto Argenta Road. This is a reasonable assumption since on both days of data collection, this fact has held true.

The saturation flow rate of the intersection is estimated to be 817 veh/hr. Therefore, this intersection is not capable of handling all show-bound vehicles coming in from the east on I-72 WB since the peak estimated flow rate of show-bound vehicles is approximately 900 veh/hr.
4.5 Functionality Analysis

Analysis of the functionality of the Traffic' Turk application is important in order to inform future deployments of the technology. Specifically, we study the data latency and the energy efficiency of the application on smart-phones.

4.5.1 Data Latency

The data latency of a Traffic' Turk data-point is defined as the time taken for a recorded data-point to be saved in the Traffic' Turk database after it has been recorded on the application. In order to be able to do any kind of real-time analysis, the data latency has to be low. However, events such as the Farm Progress Show induce a high demand on the cellphone network over which Traffic' Turk has to send its data. This causes slow transfer rates across the network. This is a known issue and Traffic' Turk has been developed with this in mind.

During the data collection at the Farm Progress Show, we specifically collected data latency numbers for each and every swipe made on the application. The idea was to be able to test the data latency in the field and get good performance. Other factors that determine the data latency is whether the counting locations are in areas where the cellphone network reception quality is good.

Some data collection associates were assigned to intersections very far away from the town, and their data is predicted to have higher data latency. The data latency also depends on whether the data collection associates have turned on the data service on their phones while collecting data. They were instructed to do so during the training session.

Figures 4.17 and 4.18 show the cumulative distribution of data saved in the database as a function of latency. In other words, these plots show how much of the data reached the database with less than a given latency in minutes.

The plots show that close to 85% of data on both days made it to the database within 3 minutes of it being collected. By the end of the day’s data
Figure 4.17: This plot shows the cumulative amount of data that has been saved in the database for a given value of latency on August 27, 2013.

Figure 4.18: This plot shows the cumulative amount of data that has been saved in the database for a given value of latency on August 28, 2013.
collection on Day 1, the database had over 90% of the data that was collected. The remainder of the data came in later on during the day when data collection associates connected to the internet on their phones.

Two data collection associates actually did not connect to the internet till after the second day of counting. On Day 2, by the end of the day’s counting, 98% of the data collected had been saved in the database. The remainder of the data was collected by manually connecting each of the phones that had not sent data to the Wi-fi network at the Macon County Highway department when data collection associates came to collect their compensation for the deployment.

These results are encouraging because a large majority of the data being collected is saved in the database relatively quickly considering that the Farm Progress Show created a very high demand for cellphone networks and that many data collection associates were in counting locations far away from the center of the town.

Possible ways to improve the data latency are to ensure that all data collection associates have their data connection switched on during the experiment. Alternatively, portable wireless hotspots can be setup in central locations or driven around by field support event managers to periodically gather data saved on the phones that are not connected to their data service.

### 4.5.2 Energy Efficiency

The energy efficiency of the application on smart-phones is an important quantity to monitor in order to assess whether the application can be deployed for long periods of time without a power source.

By design, the application is relatively energy intensive due to the requirement that the phone’s screen be switched on at all times. However, there are numerous battery life optimizations that could be performed if the energy efficiency is not up to the required standard.

The factors that influence battery life of a phone when running TrafficTurk range from the model of the phone being used to the age of a phone’s battery to the weather conditions in which the application is being run. Therefore, a large number of phones with different manufacturers and different ages of
batteries is required to reliably test the energy efficiency of the application.

As of the previous large scale TrafficTurk deployment, we had found that the application can, on average, run continuously on a fully charged phone for approximately 3 hours. This deployment at the Farm Progress Show was ideal to test the improvements that have been made to the TrafficTurk application since the last deployment.

For this test, the average starting battery life over all data collection associates was 94.08% on Day 1 and 94.25% on Day 2. The average ending battery life was 60.82% on Day 1 and 65.57% battery life on Day 2. However, many data collection associates had access to a power source during the test and therefore, the overall average battery drain is not a good indicator of energy efficiency. In fact, 19 data collection associates on Day 1 and 17 data collection associates on Day 2 used a power source to charge their phones while they were collecting data.

Therefore, we look at the average starting and ending battery lives of data collection associates who never plugged their phone into a power source for recharging and collected data for the entire 3 hour period between 7:00am and 10:00am.

The average starting battery life of data collection associates who did not plug their phones into a power source and collected data for the entire duration on Day 1 was 95.25% and the average ending battery life was 51.66%. On Day 2, these values were 95.36% and 46.21% respectively.

The highest battery drain over all data collection associates on Day 1 was 68% and on Day 2 was 70%. The lowest battery drain over all data collection associates on Day 1 was 20% and on Day 2 was 21%.

The average battery drain over three hours over both days is approximately 46% over three hours. Therefore, we estimate that the application can run on a phone continuously for over 6 hours, which is a marked improvement for the TrafficTurk application since the last deployment.
Figure 4.19: Battery life over time of a data collection associate who never plugged in their phone into a power source during a day’s data collection period.

Figure 4.20: Battery life over time of a data collection associate who plugged in their phone into a power source during a day’s data collection period.
4.6 Scalability

The scalability of the TrafficTurk system to larger deployments and larger geographical areas is important since the technology must have the capability to be easily deployed where and when it is required. This section describes the current issues in scalability and provides recommendations and plans for improving the scalability in the future.

4.6.1 Geographical scalability

One of the key features of the TrafficTurk technology is its ability to collect data on an underlying map so as to provide a geographical location for each recorded data-point. This feature allows data to be collected by individuals who do not have a formal training in transportation engineering. People do not have to enter the exact location or orientation of their recorded data before or after data collection and this makes the data collection process much simpler.

The map also provides us with knowledge of the traffic network that can be used for prediction and control of traffic in real-time, although, these techniques are currently still in development. The current underlying map that is used is an open-source, crowd-sourced map of the world called OpenStreetMap [8]. The reason we use this map is so that any discrepancies between the physical world and the digital map can be corrected in real-time. However, the challenge in geographical scalability with this map is the enormous size of the map data.

Additionally, the application must be able to download and process the required map data when a new region of the world needs traffic data collection. This is not a trivial process because the system has to first pre-process the map, then create intersections and roads within our database in a short amount of time.

Another challenge with the OpenStreetMap data is that it is constantly changing. New roads and intersections are being added and corrected every day. Therefore, we must ensure that any data collected on an old version of the map is not invalidated when a new map version is released. These technical challenges can be overcome with some amount of engineering effort on the back-end of the system.
The current proposals to solve these issues include a tiling system where the entire map of the world is divided into smaller regions called tiles. These tiles can then be quickly downloaded, pre-processed and made available to the users of the application.

The constantly changing map issue can be resolved by saving all versions of the map that data was collected on in our database and being able to fetch the required version of the map when old data is needed or processed.

4.6.2 Deployment scalability

The largest deployment of TrafficTurk so far has been approximately 120 sensors in Urbana-Champaign during the Illinois homecoming football game. The deployment at the Farm Progress Show was approximately 31 sensors. However, the eventual goal of TrafficTurk is to be able to be deployed at a much larger scale in big cities. These deployments bring with them some issues of scale that must be investigated in order to make the TrafficTurk technology viable for greater use.

Since the technology relies on humans to collect and monitor data, efficient systems to recruit, train, support and compensate data collection associates and event managers are critical. Thus far, the different strategies to recruit people that have been used include in-person recruitment at college campuses, fliers, mass emails through university list-serves, and advertisements in local newspapers and local TV and radio channels. These methods have supported deployments of up to 150 individuals. However, moving forward, we believe that strong social media campaigns can also be used to recruit people.

Training of data collection associates is always necessary when deployments are using the help of the general public to collect traffic data. These training sessions are used to reinforce the importance of safety and punctuality for the deployment. Training sessions generally accommodate about 15 people each and occur for a duration of about 45 minutes each. However, this requirement can be greatly reduced with the introduction of in-application tutorials that let the user learn how to use the application as well as guide them through the safety considerations and punctuality requirements of the deployment. This feature is something that could be incorporated into the application in the
Another bottleneck with large deployments is the ability to compensate them for their work. Currently, data collection associates have been paid in cash at a central location in the area where data was collected. For larger deployments, online payment systems such as credit to large online retailers or online payment platforms such as PayPal or Google Wallet could be set up to compensate data collection associates remotely. This system will also require the development of algorithms to definitively assess the quality of the data before compensation is made.

Finally, all deployments rely on a team of event managers to provide support to data collection associates over the phone and in the field. This team is also trained and equipped with the necessary tools to provide support such as backup phones, phone chargers, computers and cars/bicycles.

For larger deployments of TrafficTurk where the general public is being recruited to collect data, in-person support may not be feasible. However, more robust tutorials and troubleshooting help documents contained with the application and on the TrafficTurk website will ensure that data collection associates can solve common issues independently.

Moreover, a more developed monitoring system can help a small team of event managers to remotely predict issues and inform data collection associates through the application to take preventive action. For example, if an individual’s phone battery is draining at a rate that will not let them collect data for the entire data collection period, the event managers may be able to send them a push notification that alerts them to the issue and recommends a course of action.

### 4.7 Potential Future Uses of TrafficTurk

The TrafficTurk system was designed as a way to quickly and inexpensively deploy traffic sensing infrastructure when and where it is required. More specifically, TrafficTurk was designed to be able to handle large pre-planned events and unplanned events that induce atypical traffic congestion. However, it is also useful for regular traffic data collection for signal optimization, and for testing of new and old traffic infrastructure. The key feature that separates
TrafficTurk from conventional methods of data collection is the ubiquity of the devices that the system can run on and the real-time transfer of geographically and temporally encoded data to central databases. Therefore, we discuss potential future uses for TrafficTurk that leverage these features.

- Development of Signal Coordination and Timing plans for arterials require turning movement counts that can be recorded using TrafficTurk. The advantage of using TrafficTurk to do this would be the introduction of the possibility of simultaneous data collection along the entire arterial. Incorporation of real-time algorithms to create these plans will also let traffic engineers change the signal plans as soon as they have finished collecting traffic data.

- Prediction and control of traffic during atypical traffic congestion events can be done with the TrafficTurk system. For events that create atypical traffic congestion, TrafficTurk provides the ability to quickly and inexpensively collect large amounts of traffic data simultaneously. This allows prediction algorithms to use real-time data along with models to accurately predict the state of traffic. This further allows control algorithms to devise control strategies for the entire network that can be implemented in real-time. For many atypical traffic congestion events, policemen are used to hand-direct traffic or manually override the traffic signals. TrafficTurk’s prediction and control algorithms could provide policemen with recommendations on how to control the traffic at their local intersection such that the entire network is coordinated and the flow of traffic is optimized.
The Future of Traffic Monitoring and Control of Extreme Congestion Events

Extreme congestion events are an inevitable part of today’s road traffic networks in our cities. They occur due to a number of reasons and have the potential to cause great disruption to society.

So far, traffic management during these events has not been studied extensively in the academic realm due to the lack of available traffic data during these events. However, with the ever-increasing availability of data due to improving technology, the future of traffic monitoring and control during extreme congestion events looks bright. It is quite possible that in the near future, the exact trajectory of each vehicle on the roads can be taken into account in real-time to manage traffic efficiently.

However, this is not possible before a number of technical challenges such as the development of reliable models, efficient computational methods, privacy, etc. are addressed. TrafficTurk is a technology that is designed to bridge this time gap between the present and the future where detailed traffic data is readily available and can be processed efficiently to manage traffic during extreme congestion events.

So far, the TrafficTurk system comprises of a smart-phone application that can be used to collect traffic data, a back-end system that facilitates data collection and storage, and pre-processing algorithms to infer important infor-

81
mation about the traffic network.

The eventual objective of the system is to be able to estimate the current state of traffic, predict the traffic state for a time in the future and control the traffic state evolution.

Therefore, techniques to predict and control the traffic state need to be developed to be able to comprehensively manage traffic during extreme congestion events. This thesis presented a method to estimate the traffic controller strategies using inverse optimal control as a first step towards these goals.

Preliminary work has also been done to estimate the vehicle density on road segments between two TrafficTurk sensors and has been presented in this thesis, however, there is more potential to create more robust and detailed estimation algorithms for the entire road network.

Additionally, the ability to deploy TrafficTurk at large scales across the globe in an efficient and reliable manner also needs to be developed. The progress made thus far in achieving this goal was presented in this thesis along with recommendations for the future.

More broadly, understanding traffic congestion and extreme congestion events might provide insights into the design of cities and transportation networks of the future. There may be implications for the fundamental structure of urban areas and how to design them such that they support a safer, smarter and greener transportation network capable of avoiding the negative effects of traffic congestion.
Bibliography


